Rețele de calculatoare

EDIȚIA A PATRA

Andrew S. Tanenbaum

Universitatea Vrijie
Amsterdam, Olanda

©2003 Byblos srl, www.byblos.ro

Traducere:

Colectivul de coordonare:

prof. dr. ing. Valentin Cristea
prof. dr. ing. Eugenia Kalisz
prof. dr. ing. Nicolae Țăpuș

Colectivul de traducători:

as.ing. Ana Vârbănescu
stud. Corina Stratan
prep. ing. Sabina Șerbu
ing. Mihaela Negru
prep. ing. Natalia Costea
as. ing. Răzvan Rughiniș
prep. ing. Liviu Dragomirescu
stud. Octavian Udrea
stud. Bogdan Vișinescu
ing. Mihaela Neață
stud. Vlad Sima
stud. Cătălin Cârstoiu
stud. Mihai Mircea
stud. Cristi Orban
stud. Ozana Dragomir
stud. Andrei Agapi
stud. Ana Maria Oprescu
stud. Ionuț Frujină
stud. Gabi Ghiniță
stud. Paul Chiriță
ing. Raluca Busurca
stud. Vlad Panait
ing. Octavian Purdilă
stud. Radu Niculiță
stud. Cătălin Coman

Pregătire, design, producție:

Mihai Scorțaru, Claudiu Soroiu, Adrian Pop

Editată de BYBLOS s.r.l., ©2003

București, Str. Constantin Rădulescu Motru 13/42, Tel: +40-(0)21-3309281

Sub licență Pearson Education, Inc. după:

Computer Networks, 4th ed. de *Andrew S. Tanenbaum*
©2003, 1996 by Pearson Education, Inc., Prentice-Hall PTR
Upper Saddle River, New Jersey 07458

Tipărită în România, la MASTER DRUCK,

3400 Cluj-Napoca, Str. Liebknecht 2, Tel: +40-(0)264-432497

ISBN: 973-0-03000-6

Toate drepturile sunt rezervate. Nici o parte a acestei cărți nu poate fi reprodusă, într-o formă sau printr-un mijloc oarecare, fără permisiunea scrisă a editorului.

Toate numele produselor menționate aici sunt mărci înregistrate ale respectivilor proprietari.

Rețele de calculatoare

EDIȚIA A PATRA

Pentru Suzanne, Barbara, Marvin și în memoria lui Bram și a lui Sweetie π

Alte titluri de mare succes ale lui Andrew S. Tanenbaum:

Sisteme distribuite: principii şi paradigme

Această nouă carte, scrisă împreună cu Maarten van Steen, prezintă atât principiile, cât și paradigmele sistemelor distribuite moderne. În prima parte sunt tratate în detaliu principiile de comunicare, procesele, numele, sincronizarea, consistența și replicarea, toleranța la erori și securitatea. În cea de-a doua parte se trece la prezentarea unor paradigme diferite folosite pentru crearea sistemelor distribuite, inclusiv sisteme bazate pe obiecte, sisteme distribuite de fișiere, sisteme bazate pe documente și sisteme bazate pe coordonare. Sunt discutate pe larg numeroase exemple.

Sisteme de operare moderne, ediția a doua

Acest text de mare succes prezintă în detaliu principiile sistemelor de operare și le ilustrează cu ajutorul a numeroase exemple inspirate din lumea reală. După un prim capitol introductiv, următoarele cinci capitole tratează conceptele de bază: procese și fire de execuție, situații de blocare, gestiunea memoriei, operații de intrare/ieșire. Următoarele șase capitole tratează noțiuni mai avansate, incluzând sisteme multimedia, sisteme multiprocesor, securitate. La sfârșitul cărții sunt prezentate două studii de caz detaliate: UNIX/Linux și Windows 2000.

Organizarea structurată a calculatoarelor, ediția a patra

Această carte clasică, citită în lumea întreagă și ajunsă acum la cea de-a patra ediție, furnizează introducerea ideală în studiul arhitecturii calculatoarelor. Subiectul este prezentat într-o manieră ușor de înțeles începând cu prezentarea conceptelor de bază. Există un capitol dedicat începătorilor care prezintă logica digitală, urmat de capitole în care sunt prezentate microarhitectura, setul de instrucțiuni de la nivelul arhitecturii, sistemele de operare, limbajul de asamblare și arhitecturile paralele de calculatoare.

Sisteme de operare: proiectare şi implementare, ediția a doua

Acest text despre sisteme de operare, scris împreună cu Albert S. Woodhull, este singura carte ce acoperă atât principiile sistemelor de operare cât și aplicațiile acestora la un sistem real. Sunt tratate în detaliu toate subiectele tradiționale legate de sistemele de operare. În plus, principiile sunt ilustrate cu grijă de MINIX, un sistem de operare gratuit, de tip UNIX, pentru calculatoare personale. Fiecare carte conține un CD-ROM care conține sistemul MINIX complet (cod binar și sursă). Codul sursă este prezentat într-o anexă a cărții și este explicat în detaliu în text.

CUPRINS

PREFAȚĂ **XVII**

1. INTRODUCERE **1**

1.1 UTILIZĂRILE REȚELELOR DE CALCULATOARE 2

1.1.1 Aplicații comerciale 3
1.1.2 Aplicații domestice 5
1.1.3 Utilizatorii mobili 9
1.1.4 Aspecte sociale 11

1.2 HARDWARE-UL REȚELEI 13

1.2.1 Rețele locale 15
1.2.2 Rețele metropolitane 16
1.2.3 Rețele larg răspândite geografic 17
1.2.4 Rețele fără fir 19
1.2.5 Rețelele casnice (Home networks) 21
1.2.6 Inter-rețelele 23

1.3 PROGRAMELE DE REȚEA 24
1.3.1 Ierarhiile de protocoale 24
1.3.2 Probleme de proiectare a nivelurilor 28
1.3.3 Servicii orientate pe conexiuni și servicii fără conexiuni 29
1.3.4 Primitive de serviciu 31
1.3.5 Relația dintre servicii și protocoale 33

1.4 MODELE DE REFERINȚĂ 34
1.4.1 Modelul de referință OSI 34
1.4.2 Modelul de referință TCP/IP 37
1.4.3 O comparație între modelele de referință OSI și TCP 40
1.4.4 O critică a modelului și protocoalelor OSI 41
1.4.5 O critică a modelului de referință TCP/IP 43

1.5 EXEMPLE DE REȚELE 44
1.5.1 Internet 44
1.5.5 Rețele orientate pe conexiune 53
1.5.3 Ethernet 59
1.5.4 Rețele fără fir: 802.11 61

1.6 STANDARDIZAREA REȚELELOR 64
1.6.1 Who's Who în lumea telecomunicațiilor 64
1.6.2 Who's Who în lumea standardelor internaționale 66
1.6.3 Who's Who în lumea standardelor Internet 68

1.7 UNITĂȚI DE MĂSURĂ 69

1.8 RESTUL CĂRȚII ÎN REZUMAT 70

1.9 REZUMAT 71

1.10 PROBLEME 72

2. NIVELUL FIZIC 77

2.1 BAZELE TEORETICE ALE COMUNICĂRII DE DATE 77
2.1.1 Analiza Fourier 78
2.1.2 Semnalele cu bandă de frecvență limitată 78
2.1.3 Viteza maximă de transfer de date a unui canal 81

2.2 MEDII DE TRANSMISIE GHIDATĂ 82
2.2.1 Medii magnetice 82
2.2.2 Cablul torsadat 83
2.2.3 Cablu Coaxial 84
2.2.4 Fibre optice 84

2.3 COMUNICAȚIILE FĂRĂ FIR 90
2.3.1 Spectrul electromagnetic 91
2.3.2 Transmisia radio 93
2.3.3 Transmisia prin microunde 94
2.3.4 Undele infraroșii și milimetrice 97
2.3.5 Transmisia undelor luminoase 97

2.4 SATELIȚI DE COMUNICAȚIE 98
2.4.1 Sateliți geostaționari 99
2.4.2 Sateliți de altitudine medie 103
2.4.3 Sateliți de joasă altitudine 103
2.4.4 Sateliții în comparație cu fibrele optice 105

2.5 SISTEMUL TELEFONIC 107
2.5.1 Structura sistemului telefonic 107
2.5.2 Politica din domeniul telefonic 110
2.5.3 Bucla locală: Modemuri, ADSL și transmisia fără fir 112
2.5.4 Trunchiuri și multiplexare 123
2.5.5 Comutarea 132

2.6 SISTEMUL DE TELEFONIE MOBILĂ 136
2.6.1 Prima generație de telefoane mobile: Voce analogică 137
2.6.2 A doua generație de telefoane mobile: Voce digitală 141
2.6.3 A treia generație de telefoane mobile: Voce digitală și date 149

2.7 TELEVIZIUNEA PRIN CABLU 151
2.7.1 Televiziune prin antena colectivă 151
2.7.2 Internet prin cablu 152
2.7.3 Alocarea de spectru 154
2.7.4 Modemuri de cablu 155
2.7.5 Comparație între ADSL și cablu 157

2.8 REZUMAT 158

2.9 PROBLEME 159

3. NIVELUL LEGĂTURĂ DE DATE 165

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE 166
3.1.1 Servicii oferite nivelului rețea 166
3.1.2 Încadrarea 169
3.1.3 Controlul erorilor 172
3.1.4 Controlul fluxului 173

3.2 DETECTAREA ȘI CORECTAREA ERORILOR 173
3.2.1 Coduri corectoare de erori 174
3.2.2 Coduri detectoare de erori 176

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE 179
3.3.1 Un protocol simplex fără restricții 183
3.3.2 Un protocol simplu Stop-and-Wait (pas-cu-pas) 184
3.3.3 Un protocol simplex pentru un canal cu zgomote 186

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ 189
3.4.1 Un protocol cu fereastră glisantă de un bit 191
3.4.2 Un protocol de revenire cu *n* pași (Go Back n) 194
3.4.3 Un protocol cu repetare selectivă 199

3.5 VERIFICAREA PROTOCOALELOR 204
3.5.1 Modele de tip automat finit 204
3.5.2 Modele de tip rețea Petri 207

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE 209
3.6.1 HDLC - Controlul de nivel înalt al legăturii de date 209
3.6.2 Nivelul legăturii de date în Internet 212

3.7 REZUMAT 216

3.8 PROBLEME 217

4. SUBNIVELUL DE ACCES LA MEDIU 223

4.1 PROBLEMA ALOCĂRII CANALULUI 224
4.1.1 Alocarea statică a canalului în rețelele LAN și MAN 224
4.1.2 Alocarea dinamică a canalului în rețelele LAN și MAN 225

4.2 PROTOCOALE CU ACCES MULTIPLU 226
4.2.1 ALOHA 226
4.2.2 Protocoale cu acces multiplu și detecție de purtătoare 230
4.2.3 Protocoale fără coliziuni 233
4.2.4 Protocoale cu conflict limitat 235
4.2.5 Protocoale cu acces multiplu cu divizarea frecvenței 238
4.2.6 Protocoale pentru rețele LAN fără fir 241

4.3 ETHERNET 243
4.3.1 Cablarea Ethernet 244
4.3.2 Codificarea Manchester 247
4.3.3 Protocolul subnivelului MAC Ethernet 248
4.3.4 Algoritmul de regresie exponențială binară 250
4.3.5 Performanțele Ethernet-ului 251
4.3.6 Ethernetul comutat 253
4.3.7 Ethernet-ul rapid 254
4.3.8 Ethernetul Gigabit 257
4.3.9 IEEE 802.2: Controlul legăturilor logice 260
4.3.10 Retrospectiva Ethernetului 261

4.4 REȚELE LOCALE FĂRĂ FIR 262
4.4.1. Stiva de protocoale 802.11 262
4.4.2. Nivelul fizic al 802.11 263
4.4.3 Protocolul subnivelului MAC al 802.11 265
4.4.4 Formatul cadrului 802.11 269
4.4.5 Servicii 270

4.5 REȚELE FĂRĂ FIR DE BANDĂ LARGĂ 271
4.5.1 Comparație între 802.11 și 802.16 272
4.5.2 Stiva de protocoale 802.16 273
4.5.3 Nivelul fizic 802.16 274
4.5.4 Protocolul subnivelului MAC la 802.16 276
4.5.5 Structura cadrului 802.16 278

4.6 BLUETOOTH 278
4.6.1 Arhitectura Bluetooth 279
4.6.2 Aplicații Bluetooth 280
4.6.3 Stiva de protocoale Bluetooth 281
4.6.4 Nivelul Bluetooth radio 282
4.6.5 Nivelul bandă de bază Bluetooth 283
4.6.6 Nivelul L2CAP Bluetooth 284
4.6.7 Structura cadrului Bluetooth 284

4.7. COMUTAREA LA NIVELUL LEGĂTURII DE DATE 285
4.7.1 Punți de la 802.x la 802.y 287
4.7.2 Interconectarea locală a rețelelor 289
4.7.3 Punți cu arbore de acoperire 290
4.7.4 Punți aflate la distanță 292
4.7.5 Repetoare, Noduri, Punți, Comutatoare, Rutere și Porți 292
4.7.6 LAN-uri virtuale 295

4.8 REZUMAT 302

4.9 PROBLEME 303

5. NIVELUL REȚEA 309

5.1 CERINȚELE DE PROIECTARE ALE NIVELULUI REȚEA 309
5.1.1 Comutare de pachete de tip Memorează-și-Retransmite (Store-and-Forward) 310
5.1.2 Servicii furnizate nivelului transport 310
5.1.3 Implementarea serviciului neorientat pe conexiune 311
5.1.4 Implementarea serviciilor orientate pe conexiune 313
5.1.5 Comparație între subrețele cu circuite virtuale și subrețele datagramă 314

5.2 ALGORITMI DE DIRIJARE 315
5.2.1 Principiul optimalității 317
5.2.2 Dirijarea pe calea cea mai scurtă 318
5.2.3 Inundarea 320
5.2.4 Dirijare cu vectori distanță 321
5.2.5 Dirijarea folosind starea legăturilor 324
5.2.6 Dirijare ierarhică 329
5.2.7 Dirijarea prin difuzare 331
5.2.8 Dirijarea cu trimitere multiplă (multicast) 333

5.2.9 Dirijarea pentru calculatoare gazdă mobile 334
5.2.10 Dirijarea în rețele AD HOC 337
5.2.11 Căutarea nodurilor în rețele punct la punct 341

5.3 ALGORITMI PENTRU CONTROLUL CONGESTIEI 345
5.3.1 Principii generale ale controlului congestiei 347
5.3.2 Politici pentru prevenirea congestiei 348
5.3.3 Controlul congestiei în subrețelele bazate pe circuite virtuale 349
5.3.4 Controlul congestiei în subrețele datagramă 351
5.3.5 Împrăștierea încărcării 353
5.3.6 Controlul fluctuațiilor 355

5.4 CALITATEA SERVICIILOR 356
5.4.1 Cerințe 356
5.4.2 Tehnici pentru obținerea unei bune calități a serviciilor 357
5.4.3 Servicii integrate 367
5.4.4 Servicii diferențiate 370
5.4.5 Comutarea etichetelor și MPLS 372

5.5 INTERCONECTAREA REȚELELOR 374
5.5.1 Prin ce diferă rețelele 376
5.5.2 Cum pot fi conectate rețelele 377
5.5.3 Circuite virtuale concatenate 378
5.5.4 Interconectarea rețelelor fără conexiuni 379
5.5.5 Trecerea prin tunel 380
5.5.6 Dirijarea în rețele interconectate 382
5.5.7 Fragmentarea 383

5.6 NIVELUL REȚEA ÎN INTERNET 386
5.6.1 Protocolul IP 388
5.6.2 Adrese IP 391
5.5.4 Protocoale de control în Internet 401
5.5.5 Protocolul de dirijare folosit de porțile interioare: OSPF 406
5.6.5 Protocolul de dirijare pentru porți externe: BGP 411
5.6.6 Trimiterea multiplă în Internet 412
5.6.7 IP mobil 413
5.6.8 IPv6 415

5.7 REZUMAT 423

5.8 PROBLEME 423

6. NIVELUL TRANSPORT 431

6.1 SERVICIILE OFERITE DE NIVELUL TRANSPORT 431
6.1.1 Servicii furnizate nivelurilor superioare 431
6.1.2 Primitivele serviciilor de transport 433
6.1.3 Socluri Berkeley 436
6.1.4 Un exemplu de programare cu socluri: server de fișiere pentru Internet 437

6.2 NOȚIUNI DE BAZĂ DESPRE PROTOCOALELE DE TRANSPORT 441
6.2.1 Adresarea 442
6.2.2 Stabilirea conexiunii 445
6.2.3 Eliberarea conexiunii 449
6.2.4 Controlul fluxului și memorarea temporară (buffering) 453
6.2.5 Multiplexarea 457
6.2.6 Refacerea după cădere 458

6.3 UN PROTOCOL SIMPLU DE TRANSPORT 460
6.3.1 Primitivele serviciului ales ca exemplu 460
6.3.2 Entitatea de transport aleasă ca exemplu 461
6.3.3 Exemplul văzut ca un automat finit 468

6.4 PROTOCOALE DE TRANSPORT PRIN INTERNET: UDP 471
6.4.1. Introducere în UDP 471
6.4.2. Apel de procedură la distanță (Remote Procedure Call) 472
6.4.3 Protocolul de transport în timp real – Real-Time Transport Protocol 474

6.5. PROTOCOALE DE TRANSPORT PRIN INTERNET: TCP 477
6.5.1 Introducere în TCP 477
6.5.2 Modelul serviciului TCP 478
6.5.3 Protocolul TCP 480
6.5.4 Antetul segmentului TCP 481
6.5.5 Stabilirea conexiunii TCP 484
6.5.6 Eliberarea conexiunii TCP 485
6.5.7 Modelarea administrării conexiunii TCP 485
6.5.8 Politica TCP de transmisie a datelor 487
6.5.9 Controlul congestiei în TCP 490
6.5.10 Administrarea contorului de timp în TCP 493

6.5.11 TCP și UDP în conexiune fără fir 496
6.5.12 TCP Tranzacțional 498

6.6 ELEMENTE DE PERFORMANȚĂ 499
6.6.1 Probleme de performanță în rețelele de calculatoare 500
6.6.2 Măsurarea performanțelor rețelei 502
6.6.3 Proiectarea de sistem pentru performanțe superioare 504
6.6.4 Prelucrarea rapidă a TPDU-urilor 507
6.6.5 Protocoale pentru rețele gigabit 510

6.7 REZUMAT 514

6.8 PROBLEME 515

7. NIVELUL APLICAȚIE 521

7.1 DNS - SISTEMUL NUMELOR DE DOMENII 521
7.1.1 Spațiul de nume DNS 522
7.1.2 Înregistrări de resurse 524
7.1.3 Servere de nume 527

7.2 POȘTA ELECTRONICĂ 529
7.2.1 Arhitectură și servicii 530
7.2.2 Agentul utilizator 532
7.2.3 Formatele mesajelor 534
7.2.4 Transferul mesajelor 540
7.2.5 Livrarea finală 543

7.3 WORLD WIDE WEB 548
7.3.1 Aspecte arhitecturale 549
7.3.2 Documente Web statice 564
7.3.3 Documente Web dinamice 576
7.3.4 HTTP – HyperText Transfer Protocol 583
7.3.5 Îmbunătățiri ale performanței 588
7.3.6 Web-ul fără fir 593

7.4 MULTIMEDIA 602
7.4.1 Introducere in sunetele digitale 603
7.4.2 Compresia audio 605
7.4.3 Fluxuri audio 607

7.4.4 Radio prin Internet 610
7.4.5 Voce peste IP 613
7.4.6 Introducere la video 618
7.4.7 Compresia video 621
7.4.8 Video la cerere 628
7.4.9 MBone - Coloana vertebrală pentru trimitere multiplă 634

7.5 REZUMAT 637

7.6 PROBLEME 638

8. SECURITATEA REȚELELOR 645

8.1 CRIPTOGRAFIA 648
8.1.1 Introducere în criptografie 648
8.1.2 Cifrurile cu substituție 651
8.1.3 Cifrurile cu transpoziție 652
8.1.4 Chei acoperitoare 653
8.1.5 Două principii criptografice fundamentale 657

8.2 ALGORITMI CU CHEIE SECRETĂ 658
8.2.1 DES – Data Encryption Standard 660
8.2.2 AES – Advanced Encryption Standard 662
8.2.3 Moduri de cifrare 666
8.2.4 Alte cifruri 670
8.2.5 Criptanaliza 671

8.3 ALGORITMI CU CHEIE PUBLICĂ 671
8.3.1 RSA 672
8.3.2 Alți algoritmi cu cheie publică 674

8.4 SEMNĂTURI DIGITALE 674
8.4.1 Semnături cu cheie simetrică 675
8.4.2 Semnături cu cheie publică 676
8.4.3 Rezumate de mesaje 677
8.4.4 Atacul zilei de naștere 681

8.5 GESTIONAREA CHEILOR PUBLICE 682
8.5.1 Certificate 683
8.5.2 X.509 684

8.5.3 Infrastructuri cu chei publice 685

8.6 SECURITATEA COMUNICAȚIEI 688

8.6.1 IPsec 689
8.6.2 Ziduri de protecție 692
8.6.3 Rețele private virtuale 695
8.6.4 Securitatea în comunicațiile fără fir 696

8.7 PROTOCOALE DE AUTENTIFICARE 700

8.7.1 Autentificare bazată pe cheie secretă partajată 701
8.7.2 Stabilirea unei chei secrete: schimbul de chei Diffie-Hellman 705
8.7.3 Autentificarea folosind un Centru de Distribuția Cheilor 707
8.7.4 Autentificarea folosind Kerberos 709
8.7.5 Autentificarea folosind criptografia cu cheie publică 711

8.8 CONFIDENȚIALITATEA POȘTEI ELECTRONICE 712

8.8.1 PGP-Pretty Good Privacy (rom.: Confidențialitate Destul de Bună) 712
8.8.2 PEM-Privacy Enhanced Mail (Poștă cu Confidențialitate Sporită) 716
8.8.3 S/MIME 717

8.9 SECURITATEA WEB-ULUI 717

8.9.1 Pericole 718
8.9.2 Siguranța numelor 718
8.9.3 SSL – Nivelul soclurilor sigure (Secure Sockets Layer) 725
8.9.4 Securitatea codului mobil 728

8.10 IMPLICAȚII SOCIALE 730

8.10.1 Confidențialitate 731
8.10.2 Libertatea de exprimare 733
8.10.3 Dreptul de autor 736

8.11 REZUMAT 738

8.12 PROBLEME 739

9. RECOMANDĂRI DE LECTURĂ ȘI BIBLIOGRAFIE 745

9.1 SUGESTII PENTRU LECTURI VIITOARE 745

9.1.1 Lucrări introductive și generale 746
9.1.2 Nivelul fizic 747

9.1.3 Nivelul legătură de date 749
9.1.4 Subnivelul de control al accesului la mediu 750
9.1.5 Nivelul rețea 751
9.1.6 Nivelul transport 753
9.1.7 Nivelul aplicație 753
9.1.8 Securitatea rețelelor 754

9.2 BIBLIOGRAFIE ÎN ORDINE ALFABETICĂ 756

INDEX 777

PREFAȚĂ

Această carte este acum la a patra ediție. Fiecare ediție a corespuns unei etape diferite în modul de utilizare a rețelelor de calculatoare. Când a apărut prima ediție, în 1980, rețelele erau o curiozitate academică. În 1988, când a apărut a doua ediție, rețelele erau folosite de universități și de marile firme. Când a apărut a treia ediție în 1996, rețelele de calculatoare, în special Internet-ul, au devenit o realitate zilnică pentru milioane de oameni. Noutatea celei de a patra ediții o reprezintă evoluția rapidă a rețelelor fără fir, în numeroase forme.

Imaginea rețelelor de calculatoare s-a modificat radical de la ediția a treia. În anii '90 a existat o varietate de rețele de tip LAN și WAN, împreună cu stivele de protocoale aferente. În anul 2003, singura rețea de tip LAN larg utilizată ce utilizează mediul ghidat de transmisie este Ethernet și practic toate rețelele WAN existente sunt conectate la Internet. În consecință, o importantă cantitate de informație referitoare la rețelele mai vechi a fost înlăturată.

Oricum, noile realizări în domeniu sunt și ele consistente. Cel mai important progres l-au înregistrat comunicațiile fără fir, inclusiv 802.11, buclele locale de telefonie fără fir, a doua și a treia generație de rețele celulare (2G și 3G), Bluetooth, WAP, i-mode și altele. În consecință, a fost adăugată o importantă cantitate de informație despre rețelele fără fir. Un alt subiect important de actualitate este securitatea în rețele, pentru care a fost adăugat în carte un capitol separat.

Deși cap. 1 are aceeași funcție introductivă pe care o avea și în ediția a treia, cuprinsul a fost revizuit și actualizat. De exemplu, sunt prezentate introduceri despre Internet, Ethernet, rețele LAN fără fir, împreună cu istoricul și originile acestora. Sunt tratate pe scurt și rețelele pentru utilizatori casnici.

Cap. 2 a fost restructurat într-o oarecare măsură. După o scurtă introducere în principiile comunicațiilor de date, există trei secțiuni majore despre transmisii (prin medii ghidate, medii fără fir și sateliți) urmate de încă trei secțiuni cu studii de caz (rețele comutate de telefonie publică, rețele de telefonie mobilă și rețele de televiziune prin cablu). Printre noile subiecte expuse în acest capitol se numără ADSL, comunicația fără fir în bandă largă, rețele metropolitane fără fir, accesul Internet prin cablu și DOCSIS.

Cap. 3 s-a ocupat dintotdeauna cu principiile fundamentale ale protocoalelor punct-la-punct. Ideile expuse aici au rămas în vigoare timp de decenii. Ca urmare succesiunea detaliată de exemple de protocoale prezentate în acest capitol a rămas practic neschimbată de la a treia ediție.

Din contră, în zona subnivelului MAC a existat o activitate intensă în ultimii ani, așa că s-au produs multe schimbări în cap. 4. Secțiunea dedicată Ethernet-ului a fost extinsă pentru a include și Gigabit Ethernet. S-au introdus secțiuni complet noi despre LAN-uri fără fir, comunicație fără fir în bandă largă, Bluetooth și comutare la nivel de legătură de date, inclusiv MPLS.

Cap. 5 a fost de asemenea actualizat: au fost înlăturate toate materialele referitoare la ATM și au fost adăugate materiale suplimentare despre Internet. Un alt subiect important este calitatea serviciilor, cuprinzând expuneri despre servicii integrate și servicii diferențiate. Sunt prezente în acest capitol și rețelele fără fir, împreună cu o discuție despre rutarea în rețele ad-hoc. Alte aspecte noi includ NAT și rețelele de la egal la egal (peer-to-peer).

Cap. 6 tratează în continuare nivelul transport, dar și aici au avut loc unele schimbări. Printre acestea se numără un exemplu de programare a soclurilor (sockets). Sunt prezentate și comentate două programe de câte o pagină scrise în limbajul C. Aceste programe, disponibile și pe situl Web al cărții, pot fi compilate și rulate. Împreună ele furnizează o aplicație de server de fișiere sau server de Web, pentru experimentare. Alte subiecte noi includ apelul procedurilor la distanță, RTP și tranzacții/TCP.

Cap. 7, relativ la nivelul aplicație, a fost mai clar orientat. După o scurtă introducere în DNS, restul capitolului tratează trei aspecte: poștă electronică, Web și multimedia. Fiecare dintre acestea este tratată foarte detaliat. Discuția despre modul de funcționare a Web-ului se întinde acum pe mai mult de 60 de pagini, acoperind multe subiecte, printre care pagini Web statice și dinamice, HTTP, scripturi CGI, rețele cu livrare bazată pe conținut, cookies și păstrarea temporară în memoria ascunsă (cache) a Web-ului. Sunt prezente și materiale despre modul de scriere a paginilor Web moderne, cu scurte introduceri în XML, XSL, XHTML, PHP și altele, toate însoțite de exemple funcționale. Este menționat și accesul Web fără fir, cu accent asupra i-mode și WAP. Secțiunea de multimedia cuprinde acum MP3, fluxuri audio, radio prin Internet și transmisii de voce peste IP.

Securitatea rețelelor a devenit azi atât de importantă încât i s-a acordat un nou capitol însumând peste 100 de pagini. Sunt prezentate atât principii de securitate (algoritmi simetrici și algoritmi cu chei publice, semnături digitale și certificate X.509) cât și aplicații ale acestor principii (autentificare, securitatea poștei electronice și securitatea Web). Acest capitol este

atât întins ca arie de acoperire (de la criptografie cuantică până la cenzura guvernamentală) cât și bogat în detalii (de exemplu modul de funcționare al algoritmului SHA-1).

Cap. 9, conține o listă complet nouă de recomandări bibliografice, cât și o bibliografie cuprinzătoare de peste 350 de titluri. Peste 200 dintre aceste lucrări sunt scrise după anul 2000.

Cărțile despre computere conțin foarte multe acronime. Nici cartea de față nu face excepție. După ce ați terminat de citit această carte, următorii termeni ar trebui să însemne ceva pentru dumneavoastră: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML, HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW și XML. Dar nu vă îngrijorați. Ficare din acești termeni va fi cu atenție explicat înainte de a fi utilizat. Pentru a-i ajuta pe instructorii care doresc să folosească această carte ca suport de curs, autorul a pregătit o varietate de materiale auxiliare, printre care:

Un manual cu soluțiile problemelor.
Fișiere conținând toate figurile în diferite formate
Un simulator (scris în C) pentru exemplele de protocoale din Cap. 3.
O pagină de web cu link-uri către îndrumare practice, organizații, întrebări frecvente, etc

Manualul cu soluții este disponibil la Prentice Hall (dar numai pentru instructori, nu și pentru studenți). Toate celelalte materiale pot fi găsite pe situl cărții, la adresa:

http://www.prenhall.com/tanenbaum

De acolo, faceți click pe coperta cărții.

Multe persoane m-au ajutat în timpul lucrului la a patra ediție. Aș dori în mod deosebit să mulțumesc următoarelor persoane: Ross Anderson, Elizabeth Belding-Royer, Steve Bellovin, Chatschick Bisdikian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Thomas Robertazzi, Medy Sanadidi, Christian Schmutzer, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn și Maarten van Steen.

Mulțumiri speciale sunt adresate lui Trudy Levine care a demonstrat că bunicile pot face o treabă excelentă recapitulând materialul tehnic. Shivakant Mishra s-a gândit la multe dintre problemele

dificile de la sfârșitul capitolelor. Andy Dornan mi-a recomandat lecturi suplimentare pentru Cap. 9. Jan Looyen a furnizat echipamente hardware indispensabile într-un moment critic. Dr. F de Nies s-a dovedit un expert în materie de ”cut-and-paste” atunci când a fost necesar. Editorul meu de la Prentice Hall, Mary Franz m-a aprovizionat cu mai multe materiale pentru citit decât consumasem în precedenții 7 ani și m-a ajutat în numeroase alte situații.

În sfârșit, am ajuns la persoanele cele mai importante: Suzanne, Barbara și Marvin. Suzannei pentru dragoste, răbdare și prânzurile din excursiile la iarbă verde. Barbarei și lui Marvin pentru că au fost veseli și amuzanți în permanență (mai puțin atunci cand se plângeau de îngrozitoarele manuale pentru colegiu, fapt ce m-a făcut să fiu mai cu picioarele pe pământ). Vă mulțumesc.

ANDREW S. TANENBAUM

1

INTRODUCERE

Fiecare din ultimele trei secole a fost dominat de o anumită tehnologie. Secolul al XVIII-lea a fost secolul marilor sisteme mecanice care au însoțit Revoluția Industrială. Secolul al XIX-lea a fost epoca mașinilor cu aburi. În secolul XX, tehnologia cheie este legată de colectarea, prelucrarea și distribuirea informației. Printre alte realizări, am asistat la instalarea rețelelor telefonice mondiale, la invenția radioului și a televiziunii, la nașterea și creșterea nemaivăzută a industriei de calculatoare și la lansarea sateliților de comunicații.

Datorită progresului tehnologic rapid, aceste domenii converg în ritm alert, iar diferențele între colectarea, transportul, stocarea și prelucrarea informației dispar pe zi ce trece. Organizații cu sute de birouri răspândite pe o arie geografică largă se așteaptă să poată examina în mod curent printr-o simplă apăsare de buton chiar și echipamentele lor cele mai îndepărtate. Pe măsură ce posibilitățile noastre de a colecta, prelucra și distribui informația cresc tot mai mult, cererea pentru o prelucrarea și mai sofisticată a informației crește și mai rapid.

Deși industria de calculatoare este încă tânără în comparație cu alte industrii (de exemplu, construcția de automobile și transportul aerian), domeniul calculatoarelor a cunoscut un progres spectaculos într-un timp scurt. În primele decenii de existență sistemele de calcul erau foarte centralizate, de obicei în interiorul unei singure încăperi. Adesea, această încăpere avea pereți de sticlă prin care vizitatorii se puteau holba la marea minune electronică dinăuntru. O companie de mărime mijlocie sau o universitate ar fi putut avea unul sau două calculatoare, în timp ce instituțiile mari aveau cel mult câteva zeci. Ideea că, în mai puțin de 20 de ani, calculatoare la fel de puternice, mai mici decât un timbru poștal, vor fi produse pe scară largă în milioane de exemplare părea desprinsă dintr-un scenariu științifico-fantastic.

Întrepătrunderea dintre domeniul calculatoarelor și cel al comunicațiilor a avut o influență profundă asupra modului în care sunt organizate sistemele de calcul. Conceptul de „centru de calcul” -

în accepțiunea sa de încăpere unde există un calculator mare la care utilizatorii vin să-și ruleze programele - este total depășit. Vechiul model al unui singur calculator care servește rezolvării problemelor de calcul ale organizației a fost înlocuit de un model în care munca este făcută de un număr mare de calculatoare separate, dar interconectate. Aceste sisteme se numesc **rețele de calculatoare**. Proiectarea și organizarea acestor rețele reprezintă subiectul acestei cărți.

Pe parcursul cărții vom folosi termenul „rețea de calculatoare" pentru a desemna o colecție de calculatoare autonome interconectate folosind o singură tehnologie. Se spune despre două calculatoare că sunt interconectate dacă sunt capabile să schimbe informație între ele. Conectarea nu se face neapărat printr-un cablu de cupru; pot fi folosite în acest scop fibra optică, radiații infraroșii, microunde sau sateliți de comunicații. Rețelele pot fi de dimensiuni, tipuri și forme diferite, așa cum vom vedea ceva mai târziu. Deși poate să pară straniu, nici Internet-ul și nici World Wide Web-ul (rețea de întindere mondială) nu sunt rețele de calculatoare. Dacă parcurgeți cartea până la sfârșit va fi clar și de ce. Răspunsul simplist este următorul: Internet-ul nu este o singură rețea, ci o rețea de rețele, iar WWW este un sistem distribuit care funcționează peste nivelul Internet-ului.

În literatura de specialitate, se face deseori confuzie între o rețea de calculatoare și un **sistem distribuit**. Deosebirea esențială este aceea că într-un sistem distribuit, o colecție de calculatoare independente este percepută de utilizatorii ei ca un sistem coerent unic. De obicei, el are un model sau o unică paradigmă care îl reprezintă pentru utilizatori. Adesea, un modul software aflat pe nivelul superior al sistemului de operare (numit **middleware**) este responsabil pentru implementarea acestui model. Un bun exemplu de sistem distribuit arhicunoscut este chiar World Wide Web, în care totul ia în cele din urmă forma unui document (pagina Web).

Într-o rețea de calculatoare, coerenta, modelul si programele sunt absente. Utilizatorii au în fața lor mașini locale, fără nici o intenție de a face aceste stații să arate si să se comporte într-adevăr ca un sistem unic coerent. Dacă însă mașinile se deosebesc prin structurile hardware sau chiar prin sistemul de operare, acest amănunt este vizibil pentru utilizatori. Dacă un utilizator dorește să ruleze un program, el trebuie să se înregistreze pe mașina respectivă și să lucreze acolo.

De fapt, un sistem distribuit este un sistem de programe construit peste o rețea. Programele asigură rețelei un grad mare de coeziune și transparență. De aceea, diferența majoră între o rețea și un sistem distribuit nu apare la nivel de echipamente, ci de programe (în special la nivelul sistemului de operare).

Nu mai puțin adevărat este faptul că între cele două subiecte există o suprapunere considerabilă. De exemplu, atât sistemele distribuite cât și rețelele de calculatoare au nevoie să transfere fișiere. Diferența se referă la cine invocă transferul: sistemul sau utilizatorul. Deși această carte are în vedere în primul rând rețelele, multe din subiectele abordate sunt importante și în sistemele distribuite. Pentru mai multe informații despre sistemele distribuite, a se vedea (Tanenbaum și Van Steen, 2002).

1.1 UTILIZĂRILE REȚELELOR DE CALCULATOARE

Înainte de examinarea în detaliu a problemelor tehnice, merită să arătăm de ce sunt oamenii interesați de rețelele de calculatoare și la ce pot fi ele folosite. Până la urmă, dacă nimeni nu ar fi inte-

resat de rețele de calculatoare, puține rețele ar fi construite. Vom începe cu utilizările tradiționale în cadrul companiilor și pentru utilizatorii individuali, apoi ne vom deplasa spre dezvoltările recente privind utilizatorii mobili și rețelele domestice.

1.1.1 Aplicații comerciale

Multe companii au un număr semnificativ de calculatoare. De exemplu, o companie poate folosi calculatoare pentru monitorizarea producției, pentru urmărirea evoluției stocurilor, pentru calcularea statelor de plată. La început, fiecare din aceste calculatoare putea lucra izolat de celelalte, dar, la un moment dat, managerii au decis să le conecteze între ele pentru a putea extrage și corela informații despre întreaga firmă.

În termeni mai generali, subiectul se referă la **împărțirea resurselor**, iar scopul este de a face toate programele, echipamentele și în special datele disponibile pentru oricine din rețea, indiferent de localizarea fizică a resursei și a utilizatorului. Un exemplu uzual și larg răspândit este existența unui grup de utilizatori care folosesc o imprimantă comună. Nici unul dintre utilizatori nu are nevoie de propria imprimantă, iar o imprimantă performantă de volum mare, legată în rețea este, de cele mai multe ori, mai ieftină, mai rapidă și mai ușor de întreținut decât o colecție de imprimante individuale.

Cu toate acestea, probabil chiar mai importantă decât partajarea resurselor fizice, cum sunt imprimantele, scannerele, dispozitivele de inscripționat CD-uri, este partajarea informației. Orice companie mare sau medie, dar și multe dintre companiile mici sunt total dependente de informația prelucrată de calculatoare. Cele mai multe companii țin înregistrările clienților, inventarele, evidența conturilor de încasări, rapoartele financiare, informațiile despre taxe și încă multe altele numai cu ajutorul calculatorului. Dacă toate calculatoarele sale se defectează, o bancă nu mai poate funcționa mai mult de 5 minute. O fabrica modernă, cu o linie de asamblare condusă de calculator nu ar putea continua lucrul nici măcar atât. Chiar și o mică agenție de turism sau un birou de avocatură cu trei angajați sunt, în acest moment, dependente în mare măsură de rețelele de calculatoare, care le permit angajaților accesul instantaneu la informații relevante și la documente.

Pentru companiile mai mici, toate calculatoarele sunt cel mai probabil amplasate într-un singur birou sau poate într-o singură clădire, în timp ce pentru companiile mai mari calculatoarele și angajații pot fi răspândiți într-o mulțime de birouri și fabrici din diferite țări. Cu toate acestea, un agent de vânzări din New York poate avea uneori nevoie de acces la o bază de date cu inventarul produselor aflată în Singapore. Cu alte cuvinte, numai faptul ca un utilizator se află la 15.000 km de datele de care are nevoie nu îl poate împiedica să-și folosească datele ca și când ele ar fi locale. Pe scurt, scopul poate fi definit ca o încercare de a termina cu „tirania geografiei”.

În termenii cei mai simpli se poate imagina sistemul informațional al unei companii ca fiind alcătuit din una sau mai multe baze de date și un număr de angajați care au nevoie de acces de la distanță. În acest model, datele sunt memorate în calculatoare performante, numite **servere (servers)**. Adesea, acestea sunt plasate și întreținute centralizat de un administrator de sistem. Din contră, angajații au mașini mai simple, numite **clienți (clients)**, plasate pe birourile lor, prin intermediul cărora accesează datele aflate la distanță pentru a le include, de exemplu, în foile de calcul pe care le construiesc. (Uneori ne vom referi la operatorul care folosește o mașină client cu numele de „client”, dar va fi clar din context dacă referirea este la mașină sau la utilizatorul ei). Mașinile server si client sunt conectate în rețea, așa cum este ilustrat în fig. 1-1. De notat că am reprezentat rețeaua ca un simplu oval, fără nici un alt detaliu. Vom mai folosi această formă pentru a reprezenta o rețea în mod abstract. Atunci când sunt necesare mai multe detalii, ele vor fi furnizate.

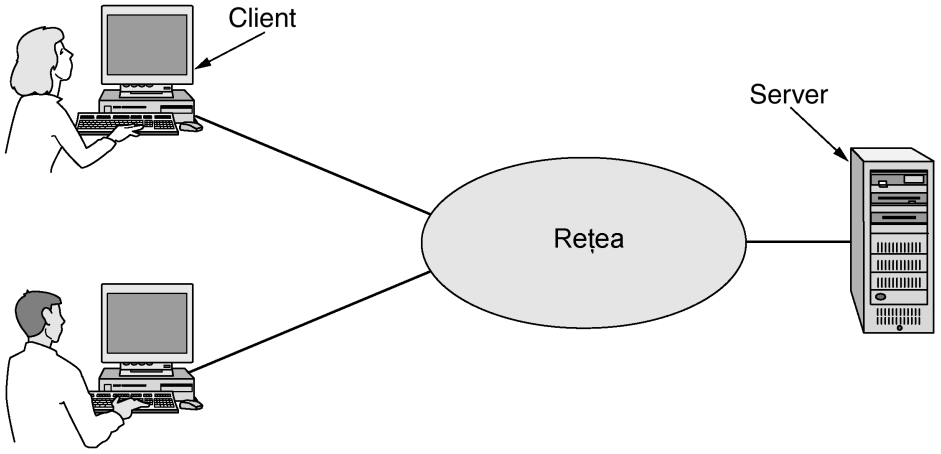


Fig. 1-1. O rețea cu doi clienți și un server.

Această structură reprezintă **modelul client-server**. Este folosit frecvent și reprezintă baza pe care lucrează multe rețele. Este aplicabil atunci când clientul și serverul se află în aceeași clădire (de exemplu, dacă ambele aparțin aceleiași companii), dar și atunci când între ele este o distanță mai mare. De exemplu, atunci când o persoană aflată acasă face un acces la o pagină Web, este folosit același model, în care serverul Web aflat la distanță are rol de server, iar calculatorul personal al utilizatorului are rol de client. În cele mai multe situații, un server poate lucra cu un număr mare de clienți.

Dacă privim mai în detaliu modelul client-server, constatăm că sunt implicate două procese, unul aflat pe mașina client și unul aflat pe mașina server. Comunicația ia forma transmiterii prin rețea a unui mesaj de la procesul client către procesul server. În continuare, procesul client va aștepta un mesaj de răspuns. Atunci când procesul server primește cererea, execută acțiunea solicitată sau caută datele cerute și transmite un răspuns. Aceste mesaje sunt prezentate în fig. 1-2.

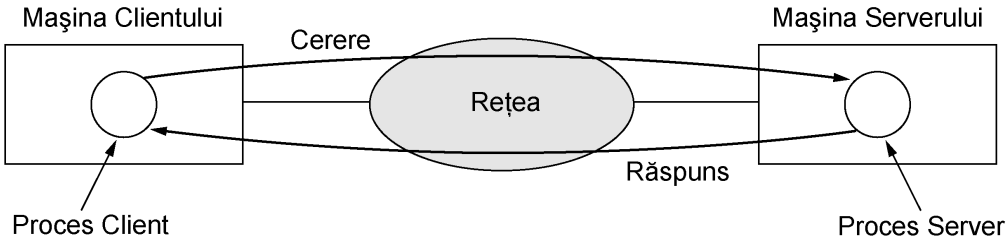


Fig. 1-2. Modelul client-server implică cereri și răspunsuri.

Un al doilea scop al construirii unei rețele de calculatoare este mai mult legat de oameni decât de informație sau chiar calculatoare. O rețea de calculatoare poate constitui un puternic **mediu de comunicare** între angajați. Aproape orice companie care are două sau mai multe calculatoare are acum **poștă electronică** (**e-mail**), pe care angajații o folosesc intens pentru comunicațiile zilnice. De fapt, una dintre neplăcerile discutate intens între angajați este multitudinea de mesaje, în mare parte lipsite de sens, cu care trebuie să se confrunte zilnic pentru că șefii au descoperit că pot trimite același mesaj (de cele mai multe ori chiar fără conținut) tuturor subordonaților, prin apăsarea unui singur buton.

Dar poșta electronică nu este singura formă de comunicație îmbunătățită care a fost făcută posibilă de rețelele de calculatoare. Cu o rețea, este ușor pentru doi oameni care lucrează la mare distanță unul de altul să scrie un raport împreună. Când unul dintre ei face o modificare asupra unui document din rețea, ceilalți vor putea vedea modificarea imediat, în loc să aștepte o scrisoare timp de mai multe zile. O astfel de accelerare face din cooperarea în cadrul grupurilor de oameni aflați la distanță o simplă comunicare, fapt imposibil cu ceva timp în urmă.

O altă formă de comunicare asistată de calculator o reprezintă videoconferințele. Folosind această tehnologie, angajații din locuri aflate la distanță pot ține o întrunire, pot să se vadă și să se audă între ei, și pot scrie chiar pe o tablă virtuală partajată. Videoconferința este o modalitate eficientă de eliminare a costurilor și timpului pierdute anterior pentru a călători. Se spune uneori că între comunicare și transport este o competiție și că activitatea care câștigă o face pe cealaltă să pară depășită.

Un al treilea scop pentru tot mai multe companii este realizarea electronică a comerțului cu alte companii, în special cu furnizorii și clienții. De exemplu, producătorii de automobile, avioane sau calculatoare, printre alții, cumpără subansamble de la diverși furnizori și apoi le asamblează. Folosind rețelele de calculatoare, producătorii pot plasa comenzile electronic, după cum este nevoie. Posibilitatea de a plasa comenzi în timp real (dacă este nevoie) reduce necesitatea stocurilor mari și sporește eficiența.

Un al patrulea scop care devine din ce în ce mai important este realizarea de tranzacții cu consumatorii prin Internet. Companiile aeriene, librăriile și magazinele de muzică au descoperit că multor consumatori le place comoditatea de a-și face cumpărăturile de acasă. În consecință, multe companii oferă on-line cataloage cu bunurile și serviciile disponibile și chiar primesc comenzi on-line. Este de așteptat ca acest sector să se dezvolte rapid în continuare. El este numit **comerț electronic (e-commerce, electronic commerce)**.

1.1.2 Aplicații domestice

În 1977 Ken Olsen era președinte al Digital Equipment Corporation, care era pe vremea aceea a doua companie în lume în vânzarea de calculatoare (după IBM). Atunci când a fost întrebat de ce Digital nu se implică mai mult în piața calculatoarelor personale, el a răspuns: „Nu există nici un motiv ca fiecare individ să aibă un calculator acasă.” Istoria a arătat că răspunsul a fost greșit, iar Digital nu mai există. De ce cumpără oamenii calculatoare pentru a le folosi acasă? La început, pentru prelucrarea de texte și pentru jocuri, dar în ultimii ani această imagine s-a schimbat radical. Probabil că în acest moment cel mai important motiv este accesul la Internet. Unele dintre cele mai populare utilizări ale Internet-ului pentru utilizatorii casnici sunt următoarele:

1. Accesul la informație de la distanță.
2. Comunicațiile interpersonale.
3. Divertismentul interactiv
4. Comerțul electronic

Accesul informației la distanță ia forme multiple. Poate fi navigarea pe Web pentru informații sau doar pentru distracție. Categoriile de informații disponibile includ artele, afacerile, gastronomia, guvernarea, sănătatea, istoria, preocupările din timpul liber, modalitățile de recreere, știința, sporturile, călătoriile, și multe altele. Distracția este de prea multe feluri ca să poată fi menționate, plus câteva care e mai bine să rămână nemenționate.

Multe ziare sunt acum disponibile on-line și pot fi personalizate. De exemplu, este uneori posibil să spui unui ziar că dorești să obții totul despre politicienii corupți, despre marile incendii, despre scandalurile în care sunt implicate celebritățile și despre epidemii, dar nu despre fotbal. Uneori este chiar posibil să vă aduceți articolele selectate pe discul local, în timp ce dormiți, sau să le tipăriți înainte de micul dejun. Și cum această tendință continuă să se dezvolte, va cauza o creștere importantă a ratei șomajului printre băieții de 12 ani care distribuie ziare, dar redacțiilor ziarelor le place această variantă, pentru că distribuția a fost întotdeauna cea mai slabă verigă din întregul lanț de producție.

Pasul următor după ziare (împreună cu revistele și jurnalele științifice) este biblioteca digitală on-line. Multe organizații profesionale, cum sunt ACM (*www.acm.org*) și IEEE Computer Society (*www.computer.org*) au deja disponibile on-line multe dintre jurnale și prezentări de la conferințe. Alte grupuri urmează rapid această tendință. În funcție de costul, dimensiunile și greutatea unui calculator portabil, cărțile tipărite vor deveni desuete. Scepticii ar trebui să fie atenți la efectul pe care l-a avut tiparul asupra manuscriselor medievale iluministe.

Toate aceste aplicații presupun interacțiuni între o persoană și o bază de date aflată la distanță. O a doua categorie largă de utilizări ale rețelei este comunicarea între persoane - este vorba în primul rând de replica secolului XXI la telefonul din secolul al XIX-lea. Poșta electronică, sau **e-mail**-ul, este deja folosită zi de zi de milioane de oameni din toată lumea și gradul de utilizare este în continuă creștere. Conține deja, în mod curent, pe lângă text și poze, secvențe audio și video. În schimb, va dura ceva mai mult până când se va pune la punct înglobarea mirosului în mesaje.

Orice adolescent este dependent de **mesageria instantanee** (**instant messaging**). Această facilitate, derivată din programul UNIX *talk* (ro: vorbește) folosit încă din anii 1970, le permite celor doi care doresc să comunice să-și trimită mesaje unul altuia în timp real. O versiune multipersonală a acestei idei este **chat-room-ul** (ro: **camera de discuții**) în care o persoană dintr-un grup poate trimite mesaje către întregul grup.

Grupurile de știri de pe tot globul, cu discuții privind orice subiect imaginabil, fac deja parte din realitatea cotidiană a unei anumite categorii de persoane, iar acest fenomen va crește până la dimensiunile întregii omeniri. Discuțiile, în care o persoană trimite un mesaj și toți ceilalți abonați ai grupului de interes pot să-l citească, se derulează în toate stilurile posibile, putând fi la fel de bine extrem de amuzante sau de pătimașe. Spre deosebire de camerele de discuții (chatroom-uri), grupurile de interese nu sunt în timp real și mesajele sunt salvate astfel încât atunci când cineva se întoarce din vacanță, toate mesajele care au fost primite între timp așteaptă cuminți să fie citite.

Un alt tip de comunicație interpersonală se numește adesea comunicație **de la egal-la-egal** (**peer-to-peer**), pentru a o distinge de modelul client-server (Parameswaran et al., 2001). În această formă, persoanele independente care formează un grup oarecare comunică în cadrul grupului, după cum se vede în fig. 1-3. Fiecare persoana poate, în principiu, să comunice cu una sau mai multe persoane; nu există o departajare clară între clienți și servere.

Comunicațiile de la egal-la-egal au explodat în jurul anului 2000 cu un serviciu numit Napster, care la apogeu avea peste 50 de milioane de fani ai muzicii care schimbau între ei melodii. A fost probabil cea mai mare înfrângere a drepturilor de autor din toată istoria lor (Lam și Tan, 2001; și Macedonia, 2000). Ideea era destul de simplă. Membrii înregistrau muzica pe care o aveau pe discurile locale într-o bază de date centrală întreținută de serverul Napster. Dacă un membru dorea o melodie, verifica baza de date ca să vadă cine o are și se ducea direct la sursă pentru a o lua. Și pentru că Napster nu ținea nici un fel de muzică pe mașinile proprii, Napster a argumentat că nu a încălcat drepturile de autor ale nimănui. Dar tribunalul nu a fost de acord și a închis sistemul.

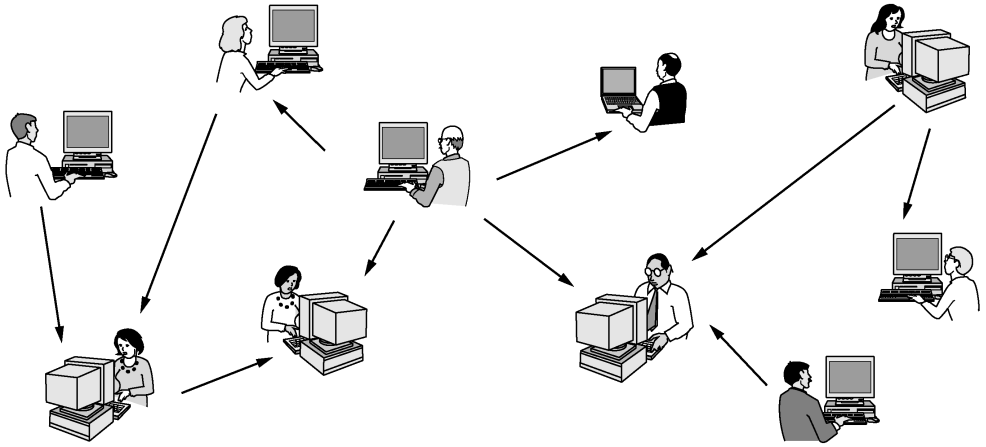


Fig. 1-3. Într-un sistem de la egal la egal nu sunt clienți și servere fixe.

Oricum, următoarea generație de sisteme de la egal-la-egal elimină baza de date centrală deoarece fiecare utilizator își va întreține propria bază locală și va oferi o listă de alți utilizatori membri ai sistemului aflați în apropiere. Un nou utilizator va putea atunci să viziteze fiecare membru și să vadă ce anume are acesta și care este lista de utilizatori aflați în apropierea sa. Acest proces de căutare poate fi repetat la infinit pentru a crea o bază de date de dimensiune mare cu ceea ce se regăsește în sistem. Este o activitate care ar deveni tracasantă pentru utilizatori, dar pentru care calculatoarele sunt excelente.

Există de asemenea și aplicații legale pentru comunicațiile de la egal-la-egal. De exemplu, fanii partajează muzica neprotejată de drepturile de autor sau noile extrase de melodii pe care formațiile muzicale le oferă în scop publicitar, familiile partajează poze, filme și informații genealogice, iar adolescenții joacă on-line jocuri cu mai mulți participanți. De fapt, una dintre cele mai populare aplicații ale Internet-ului, poșta electronică, este în mod implicit de la egal-la-egal. Este de așteptat ca această formă de comunicație să crească semnificativ în viitor.

Criminalitatea electronică nu este limitată la încălcarea drepturilor de autor. O altă zonă fierbinte este cea a jocurilor de noroc electronice. Calculatoarele au simulat tot felul de lucruri timp de decenii. De ce să nu simuleze și automatele cu fise, roata ruletei, masa de blackjack, și multe alte echipamente pentru jocurile de noroc? Ei bine, deoarece este ilegal în multe locuri. Problema este că jocurile de noroc sunt legale în multe alte părți (în Anglia, de exemplu) și proprietarii de cazinouri din astfel de state au înțeles potențialul jocurilor de noroc pe Internet. Ce se întâmplă dacă jucătorul și cazinoul se află în țări diferite, cu legi diferite? Bună întrebare.

Alte aplicații orientate pe comunicații includ utilizarea Internet-ului ca suport pentru convorbiri telefonice, conferințe video sau radio, trei domenii în plină dezvoltare. O altă aplicație este învățământul la distanță, aceasta însemnând ca poți să urmărești cursurile de la 8 dimineața fără a trebui să te dai mai întâi jos din pat. Pe termen lung, utilizarea calculatoarelor pentru a îmbunătăți comunicațiile interumane se va putea dovedi mai importantă decât oricare alte utilizări.

A treia categorie avută în vedere este divertismentul, care reprezintă o industrie uriașă, în continuă creștere. În acest domeniu aplicația de cel mai mare succes (cea care poate să influențeze tot restul) se numește video la cerere. Este plauzibil ca peste vreo zece ani să putem selecta orice film sau program de televiziune realizat vreodată în orice țară și acesta să fie imediat disponibil pe ecra-

nul nostru. Filmele noi ar putea deveni interactive: spectatorul ar fi întrebat în anumite momente ce continuare a povestirii alege (să-l ucidă MacBeth pe Duncan sau să aștepte o ocazie mai bună?), fiind prevăzute scenarii alternative pentru toate cazurile. De asemenea, televiziunea în direct s-ar putea desfășura interactiv, cu telespectatori care participă la concursuri, care aleg câștigătorul dintre concurenții preferați și așa mai departe.

Pe de altă parte, poate că nu sistemul de video la cerere, ci jocurile vor reprezenta aplicația de maxim succes. Există deja jocuri pentru mai multe persoane cu simulare în timp real, de exemplu v-ați ascuns într-o închisoare virtuală sau simulatoare de zbor în care jucătorii unei echipe încearcă să-i doboare pe cei din echipa adversă. Dacă jocurile sunt jucate cu ochelari pentru realitatea virtuală, în medii tridimensionale, în timp real și cu imagini de calitate fotografică, atunci avem un fel de realitate virtuală globală și partajată.

Cea de-a patra categorie este comerțul electronic în cel mai larg sens al cuvântului. Cumpărăturile făcute de acasă sunt deja populare și permit utilizatorilor să inspecteze on-line cataloagele a mii de companii. Unele dintre aceste cataloage vor oferi în curând posibilitatea de a obține o prezentare video imediată a oricărui produs printr-o simplă selectare a numelui produsului. După ce un client cumpără electronic un produs, dar nu poate să își dea seama cum să îl folosească, poate fi consultat departamentul de ajutor on-line.

O altă arie de interes în care comerțul electronic este deja implementat este accesul la instituțiile financiare. Mulți oameni își plătesc facturile, își administrează conturile bancare și își manevrează investițiile electronic. Acestea se vor dezvolta și mai repede de îndată ce rețelele vor deveni mai sigure.

O zonă de interes pe care nimeni nu o întrevedea ca interesantă este talciocul electronic (flea market). Licitațiile on-line de bunuri la mâna a doua au devenit o industrie uriașă. Spre deosebire de comerțul electronic tradițional, care este construit după modelul client-server, licitațiile on-line sunt mai aproape de sistemul de la egal-la-egal, un fel de consumator-la-consumator. Unele dintre aceste forme de comerț electronic au obținut porecle simpatice, plecând de la faptul că, în limba engleză, „2 (two) ” și „to” se pronunță la fel. Cele mai populare sunt prezentate în fig. 1-4.

Prescurtare	Nume întreg	Exemplu
B2C	Companie la Consumator (Bussiness to Consumer)	Comanda de cărți online
B2B	Companie la Companie (Bussiness to Bussiness)	Fabricantul de mașini comandă cauciucuri de la furnizor
G2C	Guvern la consumator (Government to Consumer)	Guvernul distribuie formularele pentru taxe în format electronic
C2C	Consumator la Consumator (Consumer to Consumer)	Licitarea de produse mâna a doua online
P2P	Punct la Punct (Peer-to-Peer)	Partajare de fișiere

Fig. 1-4. Unele forme de comerț electronic.

Fără îndoială că domeniile de utilizare pentru rețelele de calculatoare se vor dezvolta încă și mai mult în viitor și probabil că vor aborda direcții pe care acum nu le poate prevedea nimeni. La urma urmei, câți oameni ar fi crezut în 1990 că adolescenții care își scriu plictisiți mesaje pe telefoanele mobile în timp ce călătoresc cu autobuzul vor deveni o imensă sursă de bani pentru companiile de telefonie mobilă? Cu toate acestea, serviciul de mesaje scurte este extrem de profitabil.

Rețelele de calculatoare pot deveni foarte importante pentru oamenii care se află în locuri mai greu accesibile cărora le pot oferi accesul la aceleași servicii la disponibile și celor care stau în centrul orașelor. Învățământul la distanță poate afecta hotărâtor educația; universitățile vor deveni naționa-

le sau chiar internaționale. Medicina la distanță este abia la început (de exemplu monitorizarea pacienților de la distanță), dar poate să devină mult mai importantă. Dar aplicația cea mai de succes poate să fie ceva mai practică, cum ar fi folosirea unei camere digitale în frigider pentru a vedea dacă trebuie să cumperi lapte când vii acasă de la serviciu.

1.1.3 Utilizatorii mobili

Calculatoarele mobile, cum sunt portabilele sau PDA-urile (**Personal Digital Assistant**, rom: asistent digital personal) sunt unele dintre segmentele cu dezvoltarea cea mai rapidă din industria calculatoarelor. Mulți posesori ai acestor calculatoare au calculatoare la birou și doresc să fie conectați la ele chiar și când sunt plecați de acasă sau pe drum. Și cum a avea o conexiune pe fir este imposibil în mașini sau în avioane, există un interes deosebit pentru rețelele fără fir. În această secțiune vom studia pe scurt câteva dintre aplicațiile rețelelor fără fir.

De ce și-ar dori cineva o astfel de rețea? Unul dintre motivele uzuale este că obține un birou portabil. Oamenii care călătoresc mult doresc să-și poată folosi echipamentele electronice portabile pentru a trimite și pentru a primi apeluri telefonice, faxuri și poșta electronică, pentru a naviga pe Web, pentru a accesa fișiere la distanță și pentru a se putea conecta la mașini aflate la distanță. Și vor să poată face toate acestea în orice loc de pe Pământ, de pe mare sau din aer. De exemplu, în ultima vreme, la conferințele legate de calculatoare organizatorii setează o rețea locală fără fir în încăperea în care se țin conferințele. Oricine are un calculator portabil cu un modem fără fir va trebui doar să își pornească propriul calculator pentru a fi conectat la Internet, ca și cum calculatorul ar fi conectat cu un fir într-o rețea obișnuită. Similar, unele universități au instalat rețele fără fir în campus, astfel încât studenții să poată sta la umbra copacilor și să consulte catalogul bibliotecii sau să-și citească poșta electronică.

Rețelele fără fir sunt de mare valoare pentru parcurile de taximetre, camioane, vehicule utilizate pentru livrare și chiar echipe de intervenție, pentru a fi mereu în contact cu baza. De exemplu, în multe orașe șoferii de taxi sunt oameni de afaceri independenți, nu angajați ai unei companii de taximetre. În unele dintre aceste orașe, taximetrele au un ecran pe care șoferul îl poate vedea. Când sună un client, un dispecer central introduce locul de unde trebuie preluat clientul și destinația unde acesta dorește să ajungă. Această informație este afișată pe ecranul din taximetru și se generează un semnal sonor. Primul șofer care atinge un buton al ecranului este cel care preia apelul.

Rețelele fără fir sunt de asemenea importante în domeniul militar. Dacă vrei să pornești un război oriunde în lume într-un termen scurt, a conta pe infrastructura de rețea de la fața locului nu este, cel mai probabil, o idee bună. Este mai bine să o aduci pe a ta de acasă.

Deși rețelele fără fir și calculatoarele mobile sunt deseori în strânsă legătură, ele nu sunt domenii identice, după cum arată și fig. 1-5. Aici se vede diferența între **fix fără fir** și **mobil fără fir**. Chiar și calculatoarele portabile au uneori nevoie de cablu. De exemplu, dacă un călător conectează firul de la calculatorul său portabil în priza de telefon din camera de hotel, el are mobilitate, folosindu-se totuși de cablu.

Fără fir	Mobil	Aplicații
Nu	Nu	Calculatoarele staționare de pe mesele de lucru din birouri
Nu	Da	Un calculator portabil folosit într-o camera de hotel
Da	Nu	Rețelele în clădiri mai vechi, necablate
Da	Da	Biroul portabil; PDA pentru inventarul magazinului

Fig. 1-5. Combinații de rețele fără fir și echipamente mobile.

Pe de altă parte, unele calculatoare fără fir nu sunt mobile. Un exemplu important este o companie care are o clădire mai veche, necablată pentru rețea și dorește să își interconecteze calculatoarele. Instalarea unei rețele fără fir necesită doar puțin mai mult decât a cumpăra o cutie care are ceva electronică, a o despacheta și a o conecta. Totuși, această soluție poate fi mult mai ieftină decât a pune un tehnician să tragă cabluri pentru a cabla întreaga clădire.

Există, desigur, aplicații cu adevărat mobile și fără fire, de la birourile portabile până la oamenii care, intrând în magazin cu un PDA pot face inventarul. La multe aeroporturi aglomerate, oamenii care se ocupă de primirea mașinilor care au fost închiriate lucrează cu ajutorul calculatoarelor portabile fără fire. Ei introduc numărul de înmatriculare al mașinilor care sunt returnate și echipamentul portabil, care are o imprimantă atașată, apelează calculatorul principal, obține informațiile despre închiriere și tipărește pe loc factura.

Pe măsură ce tehnologiile de comunicație fără fir devin din ce în ce mai răspândite, sunt pe cale să apară tot mai multe aplicații. Să analizăm rapid unele posibilități. Aparatele de taxat fără fir pentru plata parcării au avantaje atât pentru utilizatori cât și pentru mai marii orașului. Aparatele de taxat pot să accepte cărți de credit sau de debit și să le verifice imediat prin conexiunea fără fir. Când perioada pentru care s-a plătit expiră, aparatul poate să verifice existența unei mașini în locul de parcare (va trimite un semnal înspre ea și, dacă acesta este reflectat, în spațiul respectiv se găsește o mașină) și să raporteze poliției eventuala depășire. S-a estimat că, numai la nivelul orașelor din SUA, municipalitățile ar putea obține un plus de 10 miliarde de dolari folosind această variantă (Harte et al., 2000). Mai mult, sancționarea mai riguroasă pentru parcarea ilegală va ajuta mediul înconjurător, deoarece șoferii care știu că vor fi prinși în cazul în care parchează ilegal ar putea să folosească transportul în comun.

Automatele de gustări, băuturi și alte bunuri se găsesc peste tot. Desigur, mâncarea nu ajunge în aceste automate prin puterea magiei. Periodic, cineva vine cu un camion pentru a le umple. Dacă automatele însele ar transmite printr-o conexiune fără fir un raport în fiecare zi pentru a comunica stocurile curente, șoferul camionului ar ști ce mașini trebuie re-aprovizionate și ce cantitate din fiecare produs trebuie să aducă. O astfel de informație ar duce la o planificare mai eficientă a drumului. Desigur, această informație ar putea să fie transmisă și prin liniile telefonice standard, dar soluția de a da fiecărui automat o conexiune fixă de telefon pentru un singur apel pe zi este scumpă din cauza taxei lunare fixe.

O altă zonă în care tehnologiile de conectare fără fir pot să ducă la economii sunt citirile contoarelor pentru diverse utilități. Varianta în care consumul la energie electrică, gaze, apă, și alte utilități care se regăsesc în casele oamenilor ar putea să fie raportat folosind o astfel de conexiune fără fir, nu ar mai fi nevoie să fie trimiși pe teren angajați care să se ocupe de citirea contoarelor. Similar, detectoarele de fum fără fir ar putea să sune la divizia de Pompieri în loc să facă un zgomot infernal (care este lipsit de orice valoare dacă nu este nimeni acasă). Deoarece costul dispozitivelor radio și cel al timpului de emisie scad, din ce în ce mai multe măsurători se vor face prin intermediul rețelelor fără fire.

O arie de aplicații complet diferită pentru rețelele fără fir este mult așteptata fuziune între telefoanele mobile și PDA-uri în mici calculatoare fără cablu. O primă încercare a fost făcută cu micile PDA-uri, care puteau să afișeze pagini Web simplificate pe minusculele lor ecrane. Acest sistem, numit WAP 1.0 (Wireless Application Protocol, rom: protocolul aplicațiilor fără fir) a eșuat, tocmai din cauza ecranelor prea mici, a lărgimii de bandă scăzute și a serviciilor slabe calitativ. Dar dispozitivele și serviciile mai noi vor funcționa mai bine cu WAP 2.0.

O zonă în care aceste dispozitive pot fi excelente este denumită comerț mobil (**m-commerce**) (Senn, 2000). Forța care stă în spatele acestui fenomen constă dintr-un amalgam de producători de dispozitive PDA fără fir și operatori de rețea care încearcă din răsputeri să găsească o soluție pentru a obține o bucată din plăcinta comerțului electronic. Una dintre speranțele lor este să folosească PDA-urile fără fir pentru operațiuni bancare și pentru cumpărături. O idee este utilizarea PDA-urilor ca pe un fel de portofel electronic, autorizând plățile în magazine, ca un înlocuitor pentru banii lichizi și pentru cărțile de credit. Suma cheltuită apare apoi pe factura telefonului mobil. Din punct de vedere al magazinelor, această schemă aduce un câștig prin economisirea taxelor plătite companiei de cărți de credit, taxă care poate fi de câteva procente. Desigur, acest plan poate fi dezavantajos, deoarece clienții dintr-un magazin își pot folosi PDA-urile pentru a verifica prețurile concurenței înainte de a cumpăra. Încă și mai rău, companiile de telefoane pot oferi PDA-uri cu cititoare de coduri de bare care să permită unui client să scaneze un produs dintr-un magazin și apoi să obțină instantaneu un raport detaliat despre alte locuri în care același produs se găsește și despre prețul lui.

Deoarece operatorul rețelei știe unde anume se găsește utilizatorul, unele servicii sunt în mod intenționat dependente de loc. De exemplu, poate fi posibil să afli localizarea unui magazin de cărți sau a unui restaurant chinezesc din apropiere. Hărțile mobile sunt un alt candidat. La fel sunt și prognozele meteo foarte localizate („Când o să se oprească ploaia în curtea mea din spate?”). Fără îndoială că multe alte aplicații or să apară pe măsură ce aceste dispozitive devin tot mai răspândite.

Unul dintre lucrurile importante după care comerțul mobil s-a orientat este acela că utilizatorii de telefoane mobile sunt obișnuiți să plătească pentru tot (spre deosebire de utilizatorii de Internet, care așteaptă totul gratis). Dacă un sit Internet ar impune o taxă pentru a permite utilizatorilor săi să plătească prin intermediul cărții de credit, s-ar naște o grămadă de proteste zgomotoase din partea utilizatorilor. Dacă un operator de telefonie mobilă ar permite oamenilor să plătească pentru articolele dintr-un magazin folosind telefonul și apoi le-ar fi impus o taxă pentru acest serviciu, probabil că totul ar fi fost perceput ca normal. Timpul va decide.

Ceva mai departe în timp sunt rețelele personale (personal area networks) și calculatoarele la purtător (wearable computers). IBM a dezvoltat un ceas care rulează Linux (inclusiv sistemul de ferestre X11) și are conexiune fără fir la Internet pentru a trimite și primi mesaje prin poșta electronică (Narayanaswami et al., 2002). În viitor, oamenii vor putea schimba cărți de vizită numai prin punerea ceasurilor lor față în față. Calculatoarele la purtător, fără fir, vor putea permite accesul oamenilor în încăperi securizate în același fel în care cardurile cu benzi magnetice o fac astăzi (probabil că vor lucra în combinație cu un cod PIN sau cu măsurători biometrice). Este posibil ca aceste ceasuri să fie capabile chiar să obțină informațiile relevante în vecinătatea utilizatorului (de exemplu restaurante locale). Posibilitățile sunt infinite.

Ceasurile inteligente cu radio au fost parte din spațiul nostru mental încă de când au apărut în benzile comice cu Dick Tracy în 1946. Dar praful inteligent? Cercetătorii de la Berkley au construit un calculator fără fir într-un cub cu latura de 1 mm (Warneke et al., 2001). Aplicațiile potențiale includ evidența stocurilor, pachetelor, ba chiar și a păsărelelor, rozătoarelor și insectelor.

1.1.4 Aspecte sociale

Introducerea pe scară largă a rețelelor va ridica noi probleme sociale, etice și politice. Vom menționa pe scurt câteva dintre ele; un studiu exhaustiv ar necesita cel puțin o carte. O aplicație populară a multor rețele sunt grupurile de interese sau sistemele de informare în rețea (BBS-urile), unde oa-

menii pot schimba mesaje cu persoane având preocupări similare. Atâta vreme cât este vorba de subiecte tehnice sau de pasiuni precum grădinăritul, nu sunt motive să apară multe probleme.

Problemele se ivesc în cazul grupurilor de interese care iau în discuție subiecte delicate sau extrem de disputate, cum ar fi politica, religia sau sexul. Atitudinile exprimate în cadrul acestor grupuri pot fi considerate ofensatoare de către anumiți oameni. Mai mult chiar, nu este obligatoriu ca mesajele să se limiteze la text. Fotografii color de înaltă rezoluție și chiar scurte clipuri video pot fi acum transmise cu ușurință prin rețelele de calculatoare. Unii oameni au o atitudine neutră („trăiește și lasă-mă să trăiesc"), dar alții consideră că trimiterea anumitor materiale (de exemplu, atacuri la anumite țări sau religii, pornografia etc.) este pur și simplu inacceptabilă și trebuie cenzurată. Diverse țări au diverse legi în acest domeniu, uneori chiar contradictorii. De aceea, discuțiile sunt în continuare aprinse.

Unii oameni au dat în judecată operatori de rețea, pretinzând că ei sunt responsabili pentru informația care circulă, exact ca în cazul ziarelor și revistelor. Răspunsul inevitabil este că rețeaua e ca o companie de telefoane sau ca un oficiu poștal și nu poate controla ceea ce discută utilizatorii săi. Mai mult chiar, dacă operatorii rețelei ar cenzura mesajele, atunci probabil că ei ar putea șterge orice fără a exista nici cea mai mică posibilitate de a-i da în judecată, încălcând astfel dreptul utilizatorilor la exprimare liberă. Nu este, probabil, hazardat să afirmăm că această dezbatere va continua mult timp.

O altă dispută animată are în atenție drepturile angajaților în raport cu drepturile patronilor. Multe persoane citesc și scriu poștă electronică la serviciu. Directorii unor firme au pretins că ar avea dreptul să citească și eventual să cenzureze mesajele angajaților, inclusiv mesajele trimise de la calculatoarele de acasă, după orele de program. Numai că nu toți angajații agreează această idee.

Dar chiar admițând că directorii au o astfel de putere asupra angajaților, există o relație similară și între universități și studenți? Dar între licee și elevi? În 1994 Universitatea Carnegie-Mellon a hotărât să blocheze mesajele care veneau de la grupuri de interese legate de sex pe motivul că materialele nu erau potrivite pentru minori (adică pentru cei câțiva studenți care nu aveau încă 18 ani). Disputa izvorâtă din această decizie va dura ani întregi.

Un alt subiect cheie este relația guvern-cetățean. FBI a instalat la mulți furnizori de servicii Internet un sistem care să supravegheze toate mesajele de poștă electronică care vin și pleacă în căutarea de amănunte din domeniile sale de interes (Blaze și Bellovin, 2000; Sobel, 2001 și Zacks, 2001). Sistemul a fost numit la început „Carnivore", dar din cauza publicității negative de care a avut parte a fost redenumit cu un nume care suna ceva mai inocent: DCS1000. Dar scopul lui a rămas același: de a spiona milioane de oameni în speranța că se vor găsi informații despre activități ilegale. Din păcate, al patrulea amendament al Constituției SUA interzice cercetările guvernamentale fără mandat de căutare. Dacă aceste 54 de cuvinte scrise în secolul al 18-lea au în continuare o oarecare valoare în secolul 21, tribunalele vor rămâne ocupate până în secolul 22.

Guvernul nu are monopol la amenințarea intimității cetățeanului. Sectorul privat își are și el partea lui. De exemplu, micile fișiere denumite **cookies** (**prăjiturele**) pe care programele de navigare le stochează pe calculatoarele utilizatorilor permit companiilor să urmărească activitățile utilizatorilor în cyberspace și, de asemenea, pot face ca numerele cărților de credit, numerele de asigurări sociale sau alte informații strict confidențiale să fie accesibile în Internet (Berghel, 2001).

Rețelele de calculatoare oferă posibilitatea de a trimite mesaje anonime. În anumite situații așa ceva este de dorit. De exemplu, reprezintă un mijloc pentru studenți, soldați, angajați, cetățeni de a trage un semnal de alarmă - fără teamă de represalii - în cazul comportamentului ilegal al profesorilor, ofițerilor, directorilor sau politicienilor. Pe de altă parte, în Statele Unite și în majoritatea demo-

crațiilor, legea asigură în mod explicit dreptul unei persoane acuzate de a-și chema acuzatorul în fața Curții. Acuzațiile anonime nu pot servi drept probă.

Pe scurt, rețelele de calculatoare, asemenea industriei tipografice cu 500 de ani în urmă, permit cetățenilor obișnuiți să-și lanseze opiniile prin mijloace diferite și către audiențe diferite față de cele de până acum. Această libertate nou descoperită aduce cu ea probleme nerezolvate de ordin social, politic și moral.

Odată cu binele vine și răul. Viața pare a fi construită astfel. Internetul oferă posibilitatea de a găsi repede informații, dar multe dintre ele sunt greșit informate, tendențioase sau chiar complet eronate. Sfatul medical pe care tocmai l-ați luat de pe Internet poate să vină de la un laureat al premiului Nobel sau de la un repetent din liceu. Rețelele de calculatoare au introdus de asemenea și noi tipuri de comportamente antisociale și infracționale. Transmiterea electronică a fleacurilor și gunoaielor (eng.: junk) a devenit parte din viață pentru că oamenii au colecționat milioane de adrese pe care le vând pe CD-ROM-uri așa-zișilor agenți de marketing. Mesajele care au un conținut activ (de obicei programe sau macrouri care se execută pe mașina receptorului) pot avea efecte distructive.

Furtul de identitate devine o problemă serioasă, pentru că hoții colectează destule informații despre o potențială victimă pentru a putea obține cărți de credit și alte documente în numele acesteia. În fine, posibilitatea de a transmite digital muzică și filme a deschis ușa pentru încălcarea masivă a drepturilor de autor care sunt greu de depistat și pedepsit.

Multe dintre aceste probleme puteau fi rezolvate dacă industria de calculatoare ar fi luat în serios securitatea calculatoarelor. Dacă toate mesajele erau criptate și autentificate, ar fi fost mai greu să se comită nedreptăți sau furturi. Această tehnologie este bine conturată și o vom studia în detaliu în cap. 8. Problema este că vânzătorii de hardware și aplicații software știu că introducerea unor atribute de securitate costă bani, iar cumpărătorii nu solicită astfel de atribute. Mai mult, un număr substanțial de probleme este determinat de aplicațiile care funcționează cu erori, ceea ce se întâmplă pentru că producătorii adaugă din ce în ce mai multe facilități programelor lor, ceea ce înseamnă inevitabil mai mult cod și de aceea mai multe erori. O taxă pentru noile facilități ar putea ajuta, dar ar face produsele greu de vândut în anumite segmente de piață. Plata unei despăgubiri pentru programele care funcționează eronat ar fi foarte cinstită, doar că ar duce la faliment întreaga industrie software chiar din primul an.

1.2 HARDWARE-UL REȚELEI

A venit acum timpul să ne îndreptăm atenția de la aplicațiile și problemele sociale ale interconectării (partea distractivă) la aspectele tehnice care intervin în proiectarea rețelelor (partea serioasă de lucru). Deși nu există o taxonomie general acceptată în care pot fi încadrate toate rețelele de calculatoare, sunt extrem de importante două criterii: tehnologia de transmisie și scara la care operează rețeaua. Vom examina pe rând fiecare din aceste aspecte.

În principal există două tipuri de tehnologii de transmisie care se folosesc pe scară largă. Acestea sunt:

1. Legături cu difuzare.
2. Legături punct-la-punct.

Rețelele cu difuzare au un singur canal de comunicații care este partajat de toate mașinile din rețea. Orice mașină poate trimite mesaje scurte, numite în anumite contexte **pachete**, care sunt primite de toate celelalte mașini. Un câmp de adresă din pachet specifică mașina căreia îi este adresat pachetul. La recepționarea unui pachet, o mașină controlează câmpul de adresă. Dacă pachetul îi este adresat, mașina îl prelucrează; dacă este trimis pentru o altă mașină, pachetul este ignorat.

Să considerăm, ca analogie, că cineva se află la capătul unui coridor cu multe încăperi și strigă „Watson, vino aici: Am nevoie de tine." Deși pachetul poate fi primit (auzit) de multă lume, numai Watson va răspunde. Ceilalți pur și simplu îl ignoră. Un alt exemplu ar fi un aeroport unde se anunță că toți pasagerii zborului 644 sunt rugați să se prezinte la poarta 12.

Sistemele cu difuzare permit în general și adresarea unui pachet către *toate* destinațiile, prin folosirea unui cod special în câmpul de adresă. Un pachet transmis cu acest cod este primit și prelucrat de toate mașinile din rețea. Acest mod de operare se numește **difuzare**. Unele sisteme cu difuzare suportă de asemenea transmisia la un subset de mașini, operație cunoscută sub numele de **trimitere multiplă**. Una din schemele posibile este să se rezerve un bit pentru a indica trimiterea multiplă. Restul de n - 1 biți de adresă pot forma un număr de grup. O mașină se poate „abona" la orice grup sau la toate grupurile. Un pachet trimis unui anumit grup va ajunge la toate mașinile abonate la grupul respectiv.

Prin contrast, **rețelele punct-la-punct** dispun de numeroase conexiuni între perechi de mașini individuale. Pentru a ajunge de la sursă la destinație pe o rețea de acest tip, un pachet s-ar putea să fie nevoit să treacă prin una sau mai multe mașini intermediare. Deseori sunt posibile trasee multiple, de diferite lungimi, și de aceea descoperirea drumurilor celor mai potrivite este foarte importantă. Ca o regulă generală (deși există numeroase excepții), rețelele mai mici, localizate geografic, tind să utilizeze difuzarea, în timp ce rețelele mai mari sunt de obicei punct-la-punct. Transmisiile punct la punct cu un sigur transmițător și un singur receptor sunt numite uneori și **unicasting**.

Distanța între procesoare	Procesoare localizate în acelaşi (aceeaşi)...	Exemplu
1 m	Metru pătrat	Rețea personală
10 m	Cameră	Rețea locală
100 m	Clădire	
1 km	Campus	
10 km	Oraş	Rețea metropolitană
100 km	Țară	Rețea larg răspândită geografic
1000 km	Continent	
10.000 km	Planetă	Internet-ul

Fig. 1-6. Clasificarea procesoarelor interconectate în funcție de dimensiune.

Un criteriu alternativ pentru clasificarea rețelelor este mărimea lor. În fig. 1-6 este prezentată o clasificare a sistemelor cu procesoare multiple după mărimea lor fizică. Prima categorie o reprezintă rețelele personale (personal area networks), rețele gândite pentru o singură persoană. De exemplu,

o rețea fără fir care conectează calculatorul cu perifericele sale (tastatură, imprimantă, mouse) este o rețea personală. De asemenea, un PDA care controlează aparatul auditiv al utilizatorului sau regulatorul lui de ritm cardiac se încadrează în aceeași categorie. Mai departe de aceste rețele personale sunt rețele cu domenii mai mari. Acestea pot fi împărțite în rețele locale, rețele metropolitane și rețele larg răspândite geografic. În sfârșit, prin conectarea a două sau mai multe rețele rezultă o inter-rețea. Internet-ul este un exemplu bine cunoscut de inter-rețea. Distanța este un criteriu de clasificare important, pentru că, la scări diferite, sunt folosite tehnici diferite. În această carte ne vom ocupa de rețele din toate aceste categorii. Prezentăm mai jos o scurtă introducere în subiectul echipamentelor de rețea.

1.2.1 Rețele locale

Rețelele locale (Local Area Networks), denumite în general LAN-uri, sunt rețele private localizate într-o singură clădire sau într-un campus de cel mult câțiva kilometri. Ele sunt frecvent utilizate pentru a conecta calculatoarele personale și stațiile de lucru din birourile companiilor și fabricilor, în scopul de a partaja resurse (imprimante, de exemplu) și de a schimba informații. LAN-urile se disting de alte tipuri de rețele prin trei caracteristici: (1) mărime, (2) tehnologie de transmisie și (3) topologie.

LAN-urile au dimensiuni restrânse, ceea ce înseamnă că timpul de transmisie în cazul cel mai defavorabil este limitat și cunoscut dinainte. Cunoscând această limită, este posibil să utilizăm anumite tehnici de proiectare care altfel nu ar fi fost posibile. Totodată, se simplifică administrarea rețelei.

LAN-urile utilizează frecvent o tehnologie de transmisie care constă dintr-un singur cablu la care sunt atașate toate mașinile, așa cum erau odată cablurile telefonice comune în zonele rurale. LAN-urile tradiționale funcționează la viteze cuprinse între 10 și 100 Mbps, au întârzieri mici (microsecunde sau nanosecunde) și produc erori foarte puține. LAN-urile mai noi pot opera la viteze mai mari, până la 10 Gbps. În această carte vom păstra tradiția și vom măsura vitezele de transmisie pe linii în megabiți/sec (1 Mbps reprezintă 1.000.000 biți), și gigabiți/sec (1 Gbps reprezintă 1.000.000.000 biți).

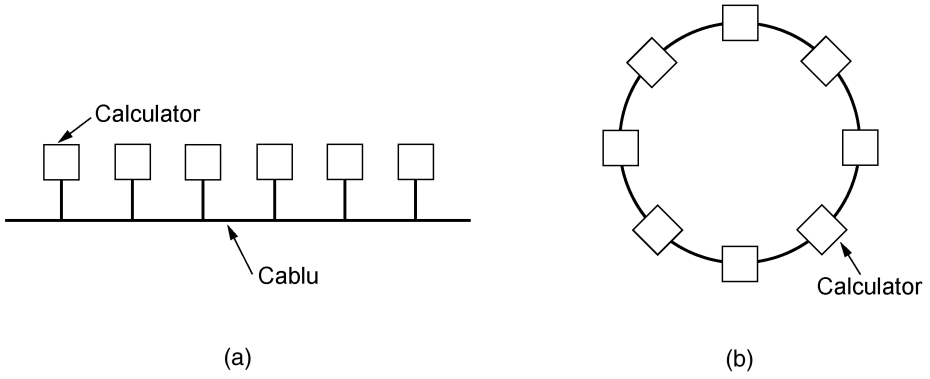


Fig. 1-7. Două rețele cu difuzare. (a) Magistrală. (b) Inel.

Pentru LAN-urile cu difuzare sunt posibile diverse topologii. Fig. 1-7 prezintă două dintre ele. Într-o rețea cu magistrală (cu cablu liniar), în fiecare moment cel mult una dintre mașini este master și are dreptul să transmită. Restul mașinilor nu pot transmite. Când două sau mai multe mașini vor

să transmită simultan, este necesar un mecanism de arbitrare. Mecanismul de arbitrare poate fi centralizat sau distribuit. De exemplu, IEEE 802.3, popular numită **Ethernet**^{TM}, este o rețea cu difuzare bazată pe magistrală cu control descentralizat, lucrând la viteze între 10 Mbps și 10 Gbps. Calculatoarele dintr-un Ethernet pot transmite oricând doresc; dacă două sau mai multe pachete se ciocnesc, fiecare calculator așteaptă o perioadă de timp aleatorie și apoi încearcă din nou.

Un al doilea tip de rețea cu difuzare este rețeaua în inel. Într-un inel fiecare bit se propagă independent de ceilalți, fără să aștepte restul pachetului căruia îi aparține. În mod tipic, fiecare bit navighează pe circumferința întregului inel într-un interval de timp în care se transmit doar câțiva biți, de multe ori înainte chiar ca întregul pachet să fi fost transmis. Ca în orice alt sistem cu difuzare, este nevoie de o regulă pentru a arbitra accesele simultane la inel. Pentru aceasta se utilizează diferite metode, care vor fi discutate în carte mai târziu. IEEE 802.5 (inelul cu jeton de la IBM) este un LAN popular de tip inel, care operează la 4 și la 16 Mbps. Un alt exemplu de rețea de tip inel este **FDDI** (**Fiber Distributed Data Interface**, rom: Interfață de date distribuite pe fibră optică).

Rețelele cu difuzare pot fi în continuare împărțite în statice și dinamice, în funcție de modul de alocare al canalului. O metodă tipică de alocare statică ar fi să divizăm timpul în intervale discrete și să rulăm un algoritm round-robin, lăsând fiecare mașină să emită numai atunci când îi vine rândul. Alocarea statică irosește capacitatea canalului atunci când o mașină nu are nimic de transmis în cuanta de timp care i-a fost alocată, astfel că majoritatea sistemelor încearcă să aloce canalul dinamic (la cerere).

Metodele de alocare dinamică pentru un canal comun sunt fie centralizate, fie descentralizate. În cazul metodei centralizate de alocare a canalului există o singură entitate, de pildă o unitate de arbitrare a magistralei, care determină cine urmează la rând. Poate face acest lucru acceptând cereri și luând o decizie conform unui algoritm intern. În cazul metodei descentralizate de alocare a canalului nu există o entitate centrală; fiecare mașină trebuie să hotărască pentru ea însăși dacă să transmită sau nu. S-ar putea crede că în acest fel se ajunge totdeauna la haos, dar lucrurile nu stau așa. Vom studia mai târziu numeroși algoritmi proiectați să refacă ordinea dintr-un potențial haos.

1.2.2 Rețele metropolitane

O **rețea metropolitană** (**Metropolitan Area Network**), sau **MAN** (plural: MAN-uri) deservește un oraș. Cel mai bun exemplu de MAN este rețeaua de televiziune prin cablu disponibilă în cele mai multe orașe. Acest sistem s-a dezvoltat de la primele antene colective folosite în zone în care semnalul recepționat prin aer era foarte slab. În aceste sisteme timpurii, o antenă foarte mare era amplasată pe vârful celui mai apropiat deal și semnalul captat era retransmis către casele abonaților.

La început, acestea erau sisteme proiectate local, ad-hoc. Apoi companiile au început să se implice în această afacere, obținând contracte de la municipalitățile orașelor pentru a cabla chiar și întreg orașul. Următorul pas a fost programarea televiziunii și chiar canale de televiziune produse numai pentru furnizarea prin cablu. De cele mai multe ori aceste canale sunt foarte specializate, pe domenii precum știrile, sporturile, gastronomia, grădinăritul, și altele. Dar încă de la începuturi și până în ultima perioadă a anilor 1990, aceste rețele erau exclusiv dedicate recepției de televiziune.

Din momentul în care Internet-ul a început să atragă audiența de masă, operatorii de rețele de cablu TV au realizat că, dacă vor face anumite schimbări în sistem, ar putea să ofere servicii bidirecționale în Internet în părțile nefolosite ale spectrului. La acel moment, sistemul de cablu TV a început să se transforme dintr-o soluție de a distribui semnalul TV în oraș într-o rețea metropolitană. La o primă aproximare, o MAN poate să arate oarecum similar cu sistemul prezentat în fig. 1-8.

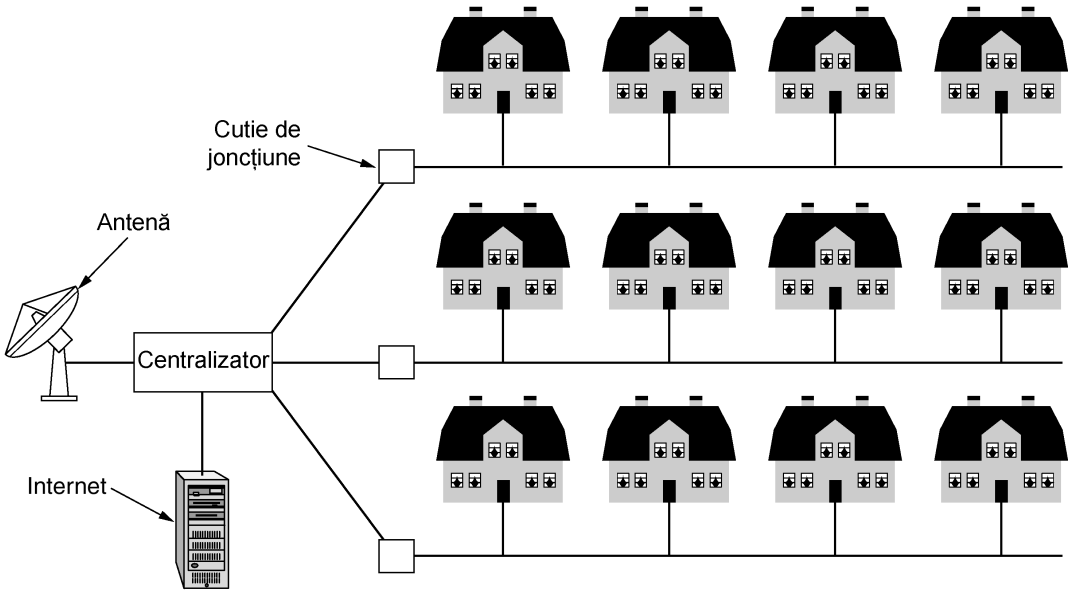


Fig. 1-8. O rețea metropolitană care se bazează pe cablu TV.

În această figură se văd atât semnalele de televiziune cât și Internet-ul trimise într-un centralizator (head end) pentru a fi apoi redistribuite în casele oamenilor. Vom reveni la acest subiect în detaliu în cap. 2.

Televiziunea prin cablu nu este singurul MAN. Ultimele dezvoltări în domeniul accesului la Internet fără fir, a dus la dezvoltarea unei noi rețele metropolitane care a fost standardizată cu numele de IEEE 802.16. Vom studia acest domeniu în cap. 2.

1.2.3 Rețele larg răspândite geografic

O **rețea larg răspândită geografic** (**Wide Area Network**), sau WAN, acoperă o arie geografică întinsă - deseori o țară sau un continent întreg. Rețeaua conține o colecție de mașini utilizate pentru a executa programele utilizatorilor (adică aplicații). În concordanță cu termenul uzual, vom numi aceste mașini **gazde**. Gazdele sunt conectate printr-o **subrețea de comunicație** sau, pe scurt, **subrețea**. Gazdele aparțin clienților (de exemplu calculatoarele personale ale oamenilor), deși subrețeaua de comunicație aparține și este exploatată, de cele mai multe ori, de o companie de telefonie sau de un furnizor de servicii Internet (ISP). Sarcina subrețelei este să transporte mesajele de la gazdă la gazdă, exact așa cum sistemul telefonic transmite cuvintele de la vorbitor la ascultător. Prin separarea aspectelor de pură comunicație ale rețelei (subrețelei) de aspectele referitoare la aplicații (gazde), proiectarea întregii rețele se simplifică mult.

În majoritatea rețelelor larg răspândite geografic, subrețeaua este formată din două componente distincte: liniile de transmisie și elementele de comutare. **Liniile de transmisie** transportă biții între mașini. Ele pot fi alcătuite din fire de cupru, fibră optică sau chiar legături radio. **Elementele de comutare** sunt calculatoare specializate, folosite pentru a conecta două sau mai multe linii de transmisie. Când sosesc date pe o anumită linie, elementul de comutare trebuie să aleagă o nouă linie pentru a retransmite datele mai departe. Din păcate, nu există nici o terminologie standard pentru de-

numirea acestor calculatoare. Aceste elemente de comutare au primit diverse nume în trecut; numele de **ruter** (**router**[1]) este acum cel mai folosit.

În acest model, prezentat în fig. 1-9, fiecare gazdă este de cele mai multe ori conectată la un LAN în care există un ruter, deși în anumite cazuri o gazdă poate fi legată direct cu un ruter. Colecția de linii de comunicație și de rutere (dar nu și gazdele) formează subrețeaua.

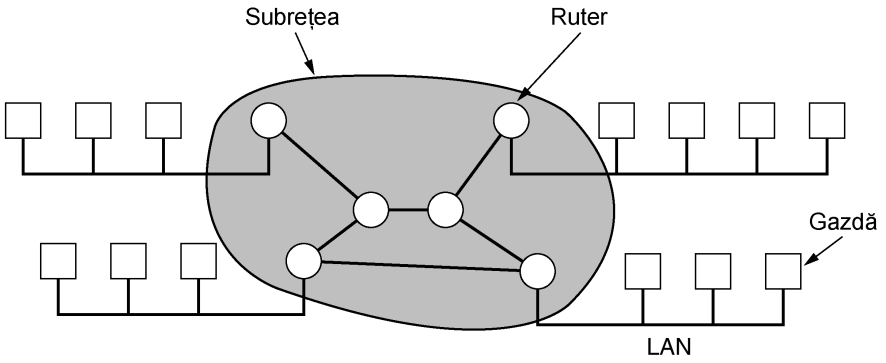


Fig. 1-9. Relația dintre gazde și subrețea.

Merită să facem un scurt comentariu în jurul termenului de „subrețea". Inițial, singura sa accepțiune se referea la colecția ruterelor și liniilor de comunicație care mutau pachetele de la gazda sursă la gazda destinație. Totuși, câțiva ani mai târziu, cuvântul a mai căpătat un al doilea înțeles, în conjuncție cu adresarea rețelelor (pe care o vom discuta în Cap. 5). Din nefericire, nu există o alternativă larg acceptată pentru înțelesul său inițial, drept care noi vom folosi acest termen, cu unele rezerve, în ambele sensuri. Din context, va fi totdeauna clar care din ele este subînțeles.

În cazul celor mai multe WAN-uri, rețeaua conține numeroase linii de transmisie, fiecare din ele legând o pereche de rutere. Dacă două rutere nu împart un același cablu, dar doresc să comunice, atunci ele trebuie să facă acest lucru indirect, prin intermediul altor rutere. Când un pachet este transmis de la un ruter la altul prin intermediul unuia sau mai multor rutere, pachetul este primit în întregime de fiecare ruter intermediar, este reținut acolo până când linia de ieșire cerută devine liberă și apoi este retransmis. O subrețea care funcționează pe acest principiu se numește subrețea **memorează-și-retransmite** sau subrețea **cu comutare de pachete**. Aproape toate rețelele larg răspândite geografic (excepție făcând cele care utilizează sateliți) au subrețele memorează-și-retransmite. Când pachetele sunt mici și au aceeași mărime, ele sunt adesea numite **celule**.

Principiul de funcționare a unui WAN cu comutare de pachete este atât de important încât merită să mai adăugăm câteva cuvinte despre el. În general, atunci când un proces al unei gazde are un mesaj de transmis către un proces de pe o altă gazdă, gazda care transmite va sparge mesajul în pachete, fiecare dintre ele reținându-și numărul de ordine din secvență. Aceste pachete sunt apoi transmise în rețea unul câte unul într-o succesiune rapidă. Pachetele sunt transportate individual prin rețea și depozitate la gazda receptoare, unde sunt reasamblate în mesajul inițial și furnizate pro-

[1] Din păcate, unii îl pronunță ca englezescul „rooter" și alții preferă să îl asocieze ca pronunție cu „doubter". Determinarea pronunției corecte în limba engleză va fi lăsată ca exercițiu cititorului. (răspunsul pe care îl veți afla poate depinde de zona în care întrebați).

cesului receptor. Un flux de pachete rezultat din descompunerea unui mesaj inițial oarecare este prezentat în fig. 1-10.

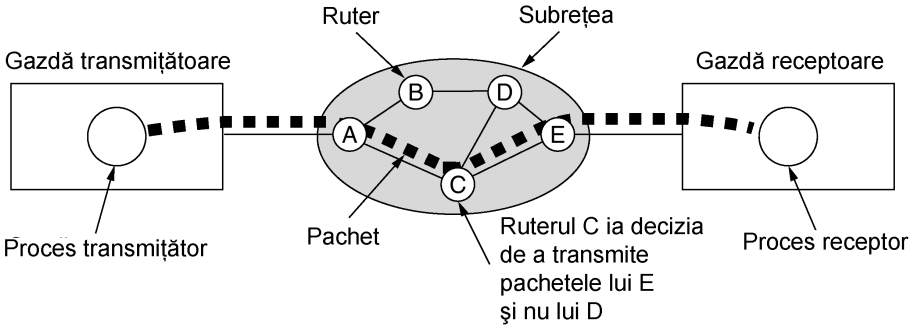


Fig. 1-10. Un flux de pachete de la transmițător la receptor.

În această figură, toate pachetele parcurg ruta A-C-E, în loc de A-B-D-E sau A-C-D-E. În unele rețele, toate pachetele aparținând unui mesaj dat trebuie să urmeze aceeași rută; în altele, fiecare pachet este dirijat separat. Desigur, dacă A-C-E este cea mai bună rută, toate pachetele pot fi transmise pe acolo, chiar dacă fiecare dintre ele este dirijat individual.

Deciziile de dirijare se iau la nivelul local al ruterului. Când un pachet ajunge la ruterul A, este de datoria lui A să decidă dacă acest pachet trebuie trimis pe linia către B sau pe linia către C. Modul în care ruterul A ia această decizie este denumit **algoritm de rutare**. Există mulți astfel de algoritmi. Pe unii dintre ei îi vom studia în detaliu în cap. 5.

Nu toate WAN-urile sunt cu comutare de pachete. O a doua posibilitate pentru un WAN este un sistem de sateliți. Fiecare ruter are o antenă prin care poate trimite și poate primi. Toate ruterele pot asculta ieșirea *de la* satelit, iar în anumite cazuri pot să asculte chiar și transmisia celorlalte rutere *către* satelit. Uneori, ruterele sunt conectate la o rețea punct-la-punct și numai unele dintre ele pot avea antene de satelit. Rețelele satelit sunt în mod implicit rețele cu difuzare și sunt foarte utile când proprietatea de difuzare este importantă.

1.2.4 Rețele fără fir

Comunicațiile digitale fără fir nu reprezintă o idee nouă. Încă din 1901, fizicianul italian Guglielmo Marconi a realizat legătura între un vapor și un punct de pe coastă folosind telegraful fără fir și codul Morse (punctele și liniile sunt, în definitiv, binare). Sistemele radio moderne au performanțe mai bune, dar ideea fundamentală a rămas aceeași.

La o primă aproximare, rețelele fără fir pot fi împărțite în 3 mari categorii:

1. Interconectarea componentelor unui sistem
2. LAN-uri fără fir
3. WAN-uri fără fir

Interconectarea componentelor se referă numai la interconectarea componentelor unui calculator folosind unde radio cu rază mică de acțiune. Aproape orice calculator are un monitor, o tastatură, un mouse și o imprimantă legate la unitatea centrală prin cabluri. Mulți dintre noii utilizatori au probleme cu conectarea tuturor cablurilor exact în mufele micuțe în care trebuie (chiar dacă acestea

sunt de cele mai multe ori codificate pe culori), așa că producătorii de calculatoare oferă opțiunea de a trimite un tehnician pentru instalare. În consecință, câteva companii s-au adunat pentru a proiecta o rețea fără fir cu rază mică de acțiune denumită Bluetooth pentru a conecta toate aceste componente fără cabluri. De asemenea, Bluetooth permite camerelor digitale, căștilor, scanerelor și altor dispozitive să se conecteze la calculator prin simpla poziționare în zona acoperită de rețea. Fără cabluri, fără instalarea de drivere, doar poziționare, pornire și ... merge. Pentru mulți oameni această ușurință în utilizare este un mare avantaj.

În cea mai simplă formă, rețelele de interconectare în sistem folosesc paradigma stăpân-sclav (master-slave) din fig. 1-11(a). Unitatea centrală a sistemului este în mod normal stăpânul, care discută cu perifericele ca sclavi. Stăpânul le comunică sclavilor ce adrese să folosească, când pot să difuzeze mesaje, cât timp pot să transmită, ce frecvențe pot să folosească, și așa mai departe. Vom discuta despre Bluetooth în detaliu în cap. 4.

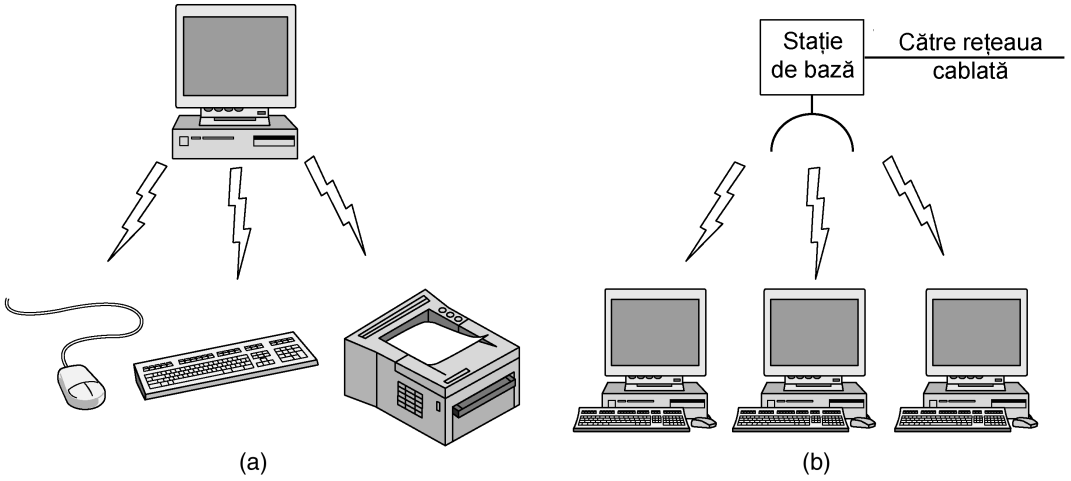


Fig. 1-11. (a) Configurație Bluetooth. (b) Rețea locală fără fir.

Următoarea treaptă în rețelele fără fir o reprezintă rețelele locale fără fir. Acestea sunt sisteme în care fiecare calculator are un modem radio și o antenă cu care poate comunica cu alte calculatoare. De multe ori există o antenă în tavan cu care mașinile vorbesc, așa cum se poate vedea în fig. 1-11(b). Oricum, dacă sistemele sunt destul de apropiate, ele pot comunica direct unul cu altul într-o configurație punct-la-punct. Rețelele locale fără fir devin din ce în ce mai utilizate în birouri mai mici și acasă, unde instalarea unei rețele Ethernet este considerată prea complicată, precum și în clădiri de birouri mai vechi, în cantinele companiilor, în camerele de conferințe, și în alte asemenea locuri. Există un standard pentru rețele locale fără fir, numit **IEEE 802.11,** pe care îl implementează majoritatea sistemelor și care devine din ce în ce mai răspândit. Îl vom discuta în cap. 4.

Cea de-a treia categorie de rețele fără fir este folosită în sistemele răspândite pe arii geografice largi (Wide Area Networks). Rețeaua radio utilizată de telefonia mobilă este un exemplu de sistem fără fir cu lărgime de bandă redusă. Acest sistem este deja la generația a treia. Prima generație era analogică și numai pentru voce. A doua generație era digitală, dar numai pentru voce. Cea de-a treia generație este digitală și este utilizată atât pentru voce cât și pentru date. Într-un anume sens, rețelele celulare fără fir sunt foarte asemănătoare cu rețelele locale fără fir, cu excepția faptului că distan-

țele implicate sunt mult mai mari, iar ratele de transfer sunt mult mai mici. Rețelele locale fără fir pot opera la rate de până la 50 Mbps pe distanțe de zeci de metri. Sistemele celulare pot opera sub 1 Mbps, dar distanțele dintre stația de bază și calculator sau telefon este măsurată mai degrabă în kilometri decât în metri. Vom avea multe de spus despre aceste rețele în cap. 2.

În plus față de aceste rețele de viteză redusă, sunt dezvoltate și WAN-uri cu lărgime de bandă mare. Important este în primul rând accesul la Internet de acasă sau din cadrul companiei prin conexiune rapidă fără fir, eliminând necesitatea folosirii sistemului de telefonie. Acest serviciu este de multe ori denumit serviciu local de distribuire multipunct. Îl vom studia mai târziu în carte. A fost dezvoltat și un standard al său, numit IEEE 802.16. Îl vom examina în cap. 4.

Aproape toate rețelele ajung mai devreme sau mai târziu să fie parte dintr-o rețea cablată pentru a oferi acces la fișiere, baze de date sau Internet. Sunt multe variante prin care aceste conexiuni pot fi realizate, în funcție de circumstanțe. De exemplu, în fig. 1-12(a) este prezentat un avion în care un număr de persoane folosesc modemuri și telefoane încorporate în spătarul scaunului (eng.: seatback telephone) pentru a suna la birou. Fiecare apel este independent de toate celelalte. O opțiune mult mai eficientă este LAN-ul zburător (flying LAN) din fig. 1-12(b). Aici, fiecare scaun este echipat cu un conector Ethernet în care pasagerii pot să își conecteze calculatoarele. Un singur ruter al avionului menține o legătură radio cu un ruter de la sol, schimbând acest ruter pe măsură ce își parcurge traseul. Această configurație este o rețea locală tradițională, doar că pentru a se conecta cu restul lumii folosește o legătură radio în loc de o linie cablată.

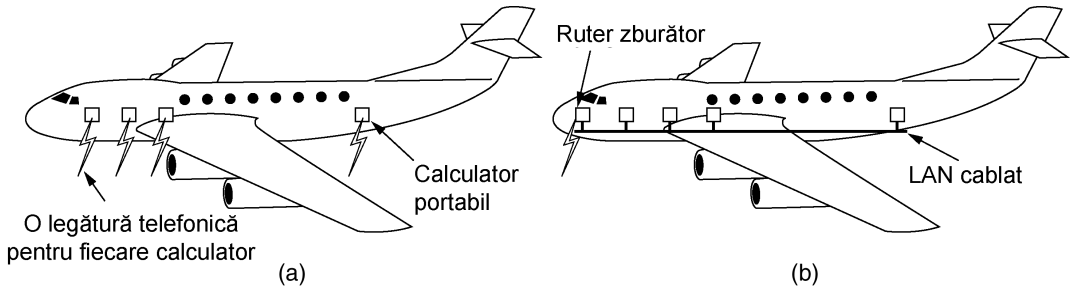


Fig. 1-12. (a) Calculatoare mobile individuale. (b) Un LAN zburător.

Multă lume crede că tehnologiile fără fir reprezintă valul viitorului (de ex. Bi et al., 2001; Leeper, 2001; Varshez și Vetter, 2000), dar exisită cel puțin o părere contrară cunoscută. Bob Metcalfe, inventatorul Ethernet-ului, a scris următoarele: „Calculatoarele mobile fără fir sunt ca băile mobile fără țevi - niște olițe de noapte portabile. Ele vor fi ceva comun în vehicule, pe șantiere și la concerte rock. Sfatul meu este să vă racordați cabluri în casă și să rămâneți acolo” (Metcalfe, 1995). Istoria ar putea să rețină această afirmație în aceeași categorie cu a lui T.J. Watson, președintele IBM, care explica în 1945 de ce IBM nu se intră în afacerea calculatoarelor: „Patru sau cinci calculatoare ar trebui să fie suficiente pentru întreaga lume până în anul 2000”.

1.2.5 Rețelele casnice (Home networks)

Rețelele în mediul casnic sunt la orizont. Ideea fundamentală este că în viitor, cele mai multe locuințe vor fi pregătite pentru instalarea de rețele. Fiecare dispozitiv din casă va fi capabil să comunice cu orice alt dispozitiv și toate vor fi accesibile prin Internet. Acesta este unul dintre acele concepte

revoluționare pe care nu l-a cerut nimeni (cum sunt telecomenzile TV sau telefoanele mobile), dar de îndată ce au fost implementate nimeni nu și-a mai putut închipui cum au trăit fără ele.

Multe dispozitive sunt capabile să fie legate în rețea. Unele dintre categoriile cele mai simple, însoțite de exemple sunt cele care urmează:

1. Calculatoarele (PC-uri staționare, PC-uri portabile, PDA-uri, periferice partajate)
2. Dispozitivele de divertisment (TV, DVD, VCR, camera video, combina muzicală)
3. Dispozitive pentru telecomunicații (telefonul, telefonul mobil, fax-ul, sistemul de comunicare interioară)
4. Aparatura casnică (cuptorul cu microunde, frigiderul, ceasul, cuptorul, aparatul de aer condiționat, luminile)
5. Contoarele și alarmele (contoarele pentru utilități, alarmele de fum sau hoți, termostatele, sistemele de supraveghere a copilului)

Rețelele casnice sunt deja implementate într-o oarecare măsură. Multe case au deja un dispozitiv pentru conectarea mai multor calculatoare la Internet printr-o conexiune rapidă. Divertismentul prin rețea nu este chiar la îndemână, dar pentru că din ce în ce mai multă muzică și mai multe filme sunt disponibile pentru descărcare din Internet, va exista o cerere de conectare a combinelor muzicale și a televizoarelor în rețea. De asemenea, oamenii vor dori să împartă propriile clipuri video cu prietenii și familia, astfel că această conexiune va trebui să fie bidirecțională. Angrenajul telecomunicațiilor este deja conectat la lumea exterioară, dar în curând aceste vor fi digitale și transmise prin Internet. În medie, o casă are cam o duzină de ceasuri (de exemplu, cele de la aparatele electrocasnice), care toate trebuie potrivite cel puțin de doua ori pe an, când se trece la ora de vară și apoi la ora de iarnă. Dacă toate aceste ceasuri ar fi conectate la Internet, această potrivire s-ar face automat. În fine, monitorizarea de la distanță a casei și a interiorului său este un posibil domeniu de succes. Probabil că mulți dintre părinți ar fi gata să cheltuiască niște bani pentru a-și supraveghea copiii adormiți, prin intermediul PDA-urilor, în timp ce iau masa în oraș, chiar și dacă au angajat un adolescent pentru a avea grijă de ei. În timp ce unii își pot imagina o rețea separată pentru fiecare zonă de aplicații, integrarea tuturor într-o singură rețea mai mare este probabil o idee mult mai bună.

Rețelele casnice au câteva proprietăți fundamental diferite de alte tipuri de rețele. Mai întâi, atât rețeaua cât și dispozitivele trebuie să fie ușor de instalat. Autorul a instalat multe componente hardware și software pe diverse calculatoare de-a lungul anilor, cu diverse rezultate. O serie de telefoane la biroul de suport tehnic al producătorului au rezultat în răspunsuri de tipul (1) Citiți manualul, (2) Reporniți calculatorul, (3) Scoateți toate componentele hardware și software cu excepția celor furnizate de noi și încercați din nou, (4) Descărcați cea mai nouă versiune a programului de configurare de pe situl nostru Web și dacă toate acestea eșuează, (5) Reformatați discul și apoi reinstalați Windows de pe CD-ROM. A spune unui cumpărător de frigider care poate fi conectat la Internet să descarce și să instaleze o nouă versiune a sistemului de operare pentru frigiderul său nu este de natură să facă prea mulți clienți fericiți. Utilizatorii de calculatoare sunt obișnuiți cu instalarea de produse care nu merg din prima; cumpărătorii de mașini, televizoare sau frigidere sunt mai puțin toleranți. Ei se așteaptă ca produsele să răspundă corect la 100% din comenzi.

În al doilea rând, rețelele și dispozitivele trebuie să fie protejate împotriva utilizării neglijente. Primele aparate de aer condiționat aveau un buton cu patru poziții: OPRIT, SCĂZUT, MEDIU, RAPID. Acum au manuale de 30 de pagini. De îndată ce vor fi conectate în rețea, așteptați-vă ca numai capitolul de securizare să aibă 30 de pagini. Ceea ce va depăși capacitatea de înțelegere a majorității utilizatorilor.

În al treilea rând, prețul scăzut este esențial pentru succes. Cumpărătorii nu vor plăti 50 de dolari în plus pentru un termostat numai pentru că unii oameni consideră important să-și supravegheze de la birou temperatura din casă. Pentru numai 5 dolari în plus, s-ar putea să se vândă.

În al patrulea rând, programul principal este foarte probabil să implice facilități multimedia, așa că rețeaua are nevoie de capacitate suficientă. Nu există piață pentru televizoare conectate la Internet care să prezinte filme de groază în rezoluție de 320 × 240 pixeli și la 10 cadre/s. Ethernet-ul rapid (fast Ethernet), mediul de lucru în majoritatea birourilor, nu este destul de bun pentru facilitățile multimedia. În consecință, rețelele casnice vor avea nevoie de performanțe mai bune decât cele are rețelelor care există acum în companii și de prețuri mai mici pentru a deveni articole care se vând în masă.

În cel de-al cincilea rând, trebuie să fie posibil să se pornească cu unul sau două dispozitive și extinderea să se poată face gradat. Aceasta înseamnă fără schimbări revoluționare. A spune consumatorilor să își cumpere periferice cu interfețe IEEE 1394 (FireWire) și apoi, după câțiva ani, să retractezi spunând că USB 2.0 este interfața lunii va face consumatorii să devină capricioși. Interfața de rețea va trebui să rămână stabilă pentru mulți ani; cablajul (dacă există) va trebui să rămână același pentru decade întregi.

În cel de-al șaselea rând, securitatea și siguranța vor fi foarte importante. Pierderea câtorva fișiere datorită unui virus de poștă electronică e una, dar dacă un hoț îți dezarmează sistemul de securitate al locuinței de la PDA-ul său și apoi intră în casă este cu totul altă situație.

O întrebare interesantă este dacă rețelele casnice trebuie să fie cablate sau fără fir. Majoritatea locuințelor au deja șase rețele instalate: electrică, telefonică, televiziune prin cablu, apă, gaz și canalizare. Adăugarea unei a șaptea rețele în timpul construcției nu este dificilă, dar reamenajarea caselor deja construite este costisitoare. Costul este un motiv de a alege rețelele fără fir, dar securitatea este un motiv pentru cele cablate. Problema cu rețelele fără fir este aceea că undele radio pe care le folosesc trec foarte ușor prin garduri. Nimeni nu este foarte bucuros dacă vecinii îi pot intercepta conexiunea la Internet și îi pot citi mesajele de poștă electronică în timp ce acestea sunt trimise la imprimantă. În cap. 8 vom vedea cum se poate folosi criptarea pentru a oferi securitate, dar în contextul unei rețele casnice, securitatea trebuie să fie și ea protejată împotriva utilizării neglijente, chiar și în cazul utilizatorilor fără experiență. Aceasta este mai ușor de spus decât de făcut, chiar și pentru utilizatori foarte pricepuți. Pe scurt, rețelele casnice oferă multe facilități și provocări. Multe dintre ele sunt legate de necesitatea de a fi ușor de administrat, sigure și securizate, mai ales în mâinile utilizatorilor care nu sunt implicați în domeniul tehnic, concomitent cu necesitatea de a obține performanțe ridicate la prețuri scăzute.

1.2.6 Inter-rețelele

În lume există multe rețele, cu echipamente și programe diverse. Persoanele conectate la o anumită rețea doresc adesea să comunice cu persoane racordate la alta. Această cerință impune conectarea unor rețele diferite, de multe ori incompatibile, ceea ce uneori se realizează utilizând mașini numite **porți** (**gateways**). Acestea realizează conectarea și asigură conversiile necesare, atât în termeni de hardware cât și de software. O colecție de rețele interconectate este numită **inter-rețea** sau **internet**. Acești termeni vor fi folosiți în sens generic, spre deosebire de Internet-ul mondial (care este un internet special), al cărui nume va fi scris mereu cu majusculă.

O formă comună de inter-rețea este o colecție de LAN-uri conectate printr-un WAN. De fapt, dacă am înlocui eticheta „subrețea” din fig. 1-9 prin „WAN”, în figură nu ar mai trebui schimbat nimic altceva. În acest caz, singura diferență tehnică reală între o subrețea și un WAN se referă la

prezența gazdelor. Dacă sistemul din interiorul zonei gri conține numai rutere, atunci este o subrețea. Dacă el conține atât rutere, cât și gazde cu utilizatori proprii, atunci este un WAN. Diferențele reale sunt legate de proprietate și utilizare.

Deseori se produc confuzii între subrețele, rețele și inter-rețele. Termenul de subrețea este mai potrivit în contextul unei rețele larg răspândite geografic, unde se referă la colecția de rutere și linii de comunicație aflate în proprietatea operatorului de rețea. Ca o analogie, sistemul telefonic constă din centrale telefonice de comutare, care sunt conectate între ele prin linii de mare viteză și sunt legate la locuințe și birouri prin linii de viteză scăzută. Aceste linii și echipamente, deținute și întreținute de către compania telefonică, formează subrețeaua sistemului telefonic. Telefoanele propriuzise (care corespund în această analogie gazdelor) nu sunt o parte a subrețelei. Combinația dintre o subrețea și gazdele sale formează o rețea. În cazul unui LAN, rețeaua este formată din cablu și gazde. Aici nu există cu adevărat o subrețea.

O inter-rețea se formează atunci când se leagă între ele rețele diferite. Din punctul nostru de vedere, legarea unui LAN și a unui WAN sau legarea a două LAN-uri formează o inter-rețea, dar nu există un consens asupra terminologiei din acest domeniu. O regulă simplă este aceea că dacă diferite companii sunt plătite să construiască diverse părți ale unei rețele și fiecare trebuie să își întrețină propria parte, avem o inter-rețea mai degrabă decât o singură rețea. De asemenea, dacă tehnologiile diferă în diverse zone ale rețelei (de exemplu: difuzare și punct-la-punct), probabil că discutăm nu despre una ci despre două rețele.

1.3 PROGRAMELE DE REȚEA

În proiectarea primelor rețele de calculatoare, s-a acordat atenție în primul rând echipamentelor, iar programele au fost gândite ulterior. Această strategie nu mai este valabilă. Programele de rețea sunt acum foarte structurate. În secțiunile următoare vom examina unele detalii ale tehnicii de structurare a programelor. Metoda descrisă aici formează punctul de sprijin al întregii cărți și ea va apărea mai departe în repetate rânduri.

1.3.1 Ierarhiile de protocoale

Pentru a reduce din complexitatea proiectării, majoritatea rețelelor sunt organizate sub forma unei serii de **straturi** sau **niveluri**, fiecare din ele construit peste cel de dedesubt. Numărul de niveluri, numele fiecărui nivel, conținutul și funcția sa variază de la rețea la rețea. Oricum, în toate rețelele, scopul fiecărui nivel este să ofere anumite servicii nivelurilor superioare, protejându-le totodată de detaliile privitoare la implementarea efectivă a serviciilor oferite. Într-un anumit sens, fiecare nivel este un fel de mașină virtuală, oferind anumite servicii nivelului de deasupra lui.

Nivelul *n* de pe o mașină conversează cu nivelul *n* de pe altă mașină. Regulile și convențiile utilizate în conversație sunt cunoscute sub numele de **protocolul** nivelului *n.* În principal, un protocol reprezintă o înțelegere între părțile care comunică, asupra modului de realizare a comunicării. Ca o analogie, atunci când o femeie este prezentată unui bărbat, ea poate hotărî să-i întindă bărbatului mâna. La rândul său, bărbatul poate decide fie să-i strângă, fie să-i sărute mâna, decizie care depinde, să spunem, dacă femeia este o avocată americană care a venit la o întâlnire de afaceri sau este o

prințesă europeană prezentă la un bal. Încălcarea protocolului va face comunicarea mai dificilă, dacă nu chiar imposibilă.

În fig. 1-13 este ilustrată o rețea cu cinci niveluri. Entitățile din niveluri corespondente de pe mașini diferite se numesc **egale**. Entitățile egale pot fi procese, dispozitive hardware, sau chiar ființe umane. Cu alte cuvinte, entitățile egale sunt cele care comunică folosind protocolul.

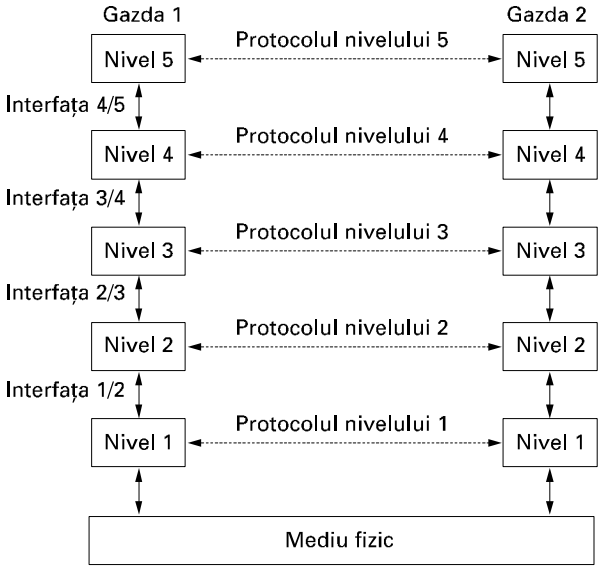


Fig. 1-13. Niveluri, protocoale și interfețe.

În realitate, nici un fel de date nu sunt transferate direct de pe nivelul *n* al unei mașini pe nivelul *n* al altei mașini. Fiecare nivel transferă datele și informațiile de control nivelului imediat inferior, până când se ajunge la nivelul cel mai de jos. Sub nivelul 1 se află **mediul fizic** prin care se produce comunicarea efectivă. În fig. 1-13, comunicarea virtuală este reprezentată prin linii punctate, iar comunicarea fizică prin linii continue. Între două niveluri adiacente există o **interfață**. Interfața definește ce operații și servicii primitive oferă nivelul de jos către nivelul de sus. Când proiectanții de rețea decid câte niveluri să includă într-o rețea și ce are de făcut fiecare din ele, unul din considerentele cele mai importante se referă la definirea de interfețe clare între niveluri.

Aceasta presupune ca, la rândul său, fiecare nivel să execute o colecție specifică de funcții clar definite. Pe lângă minimizarea volumului de informații care trebuie transferate între niveluri, interfețele clare permit totodată o mai simplă înlocuire a implementării unui nivel cu o implementare complet diferită (de exemplu, toate liniile telefonice se înlocuiesc prin canale de satelit). Așa ceva este posibil, pentru că tot ceea ce i se cere noii implementări este să furnizeze nivelului superior exact setul de servicii pe care îl oferea vechea implementare. De altfel, este un fapt obișnuit ca doua gazde să folosească implementări diferite.

O mulțime de niveluri și protocoale este numită **arhitectură de rețea**. Specificația unei arhitecturi trebuie să conțină destule informații pentru a permite unui proiectant să scrie programele sau să construiască echipamentele necesare fiecărui nivel, astfel încât nivelurile să îndeplinească corect protocoalele corespunzătoare. Nici detaliile de implementare și nici specificațiile interfețelor nu fac parte din arhitectură, deoarece acestea sunt ascunse în interiorul mașinilor și nu sunt vizibile din afară. Nu este necesar nici măcar ca interfețele de pe mașinile dintr-o rețea să fie aceleași - cu condi-

ția, însă, ca fiecare mașină să poată utiliza corect toate protocoalele. O listă de protocoale utilizate de către un anumit sistem, câte un protocol pentru fiecare nivel, se numește **stivă de protocoale.** Arhitecturile de rețea, stivele de protocoale și protocoalele propriu-zise constituie principalele subiecte ale acestei cărți.

O analogie poate ajuta la explicarea ideii de comunicare multinivel. Imaginați-vă doi filosofi (procesele egale de la nivelul 3), unul din ei vorbind limbile urdu și engleză, iar celălalt vorbind chineza și franceza. Deoarece filosofii nu cunosc o limbă comună, fiecare din ei angajează câte un translator (procesele egale de la nivelul 2), iar fiecare translator contactează la rândul său o secretară (procesele egale de la nivelul 1). Filosoful 1 dorește să comunice partenerului afecțiunea sa pentru *oryctolagus cuniculus*. Pentru aceasta, el trimite un mesaj (în engleză) prin interfața 2/3 către translatorul său, căruia îi spune următoarele cuvinte: „I like rabbits”[2] (ceea ce este ilustrat în fig. 1-14). Translatorii s-au înțeles asupra unei limbi neutre, olandeza, așa că mesajul este convertit în „Ik vind konijnen leuk.” Alegerea limbii reprezintă protocolul nivelului 2 și este la latitudinea proceselor pereche de pe acest nivel.

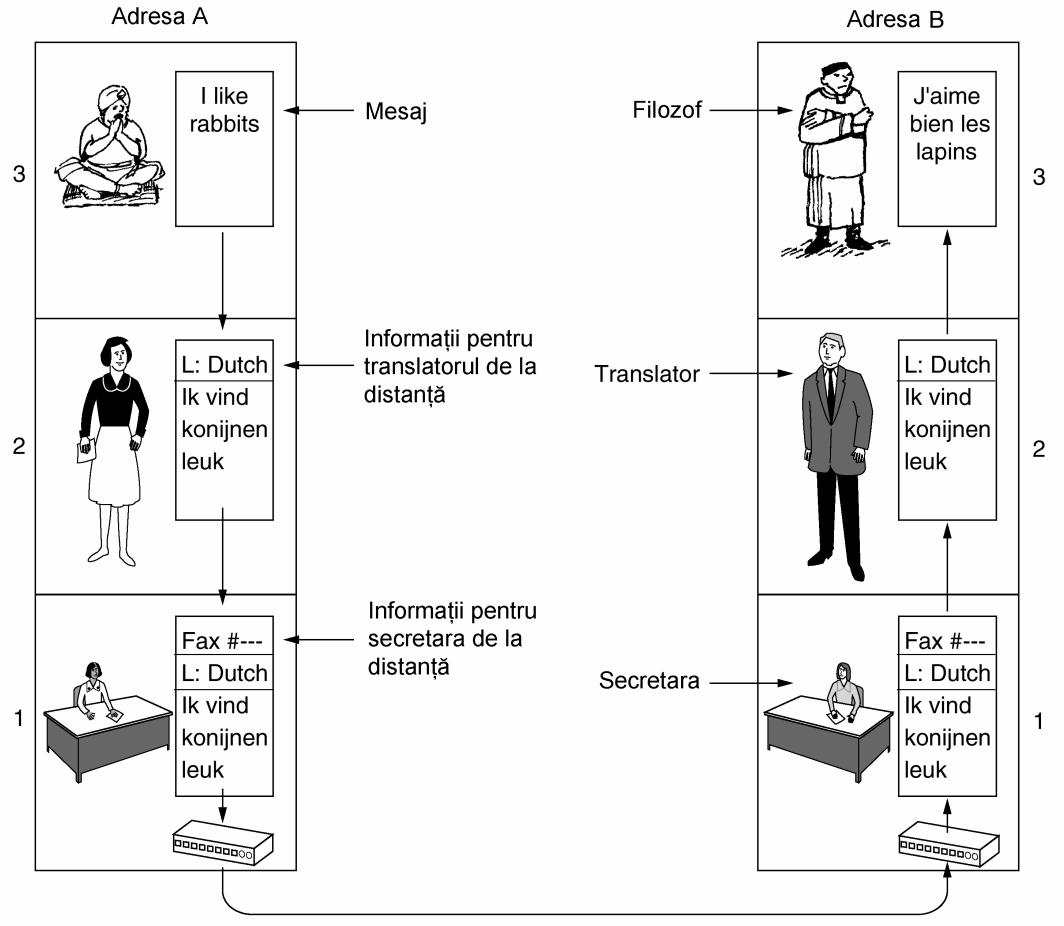


Fig. 1-14. Arhitectura filosof-translator-secretară.

[2]Propoziția înseamnă “Îmi plac iepurii.” (n.t.)

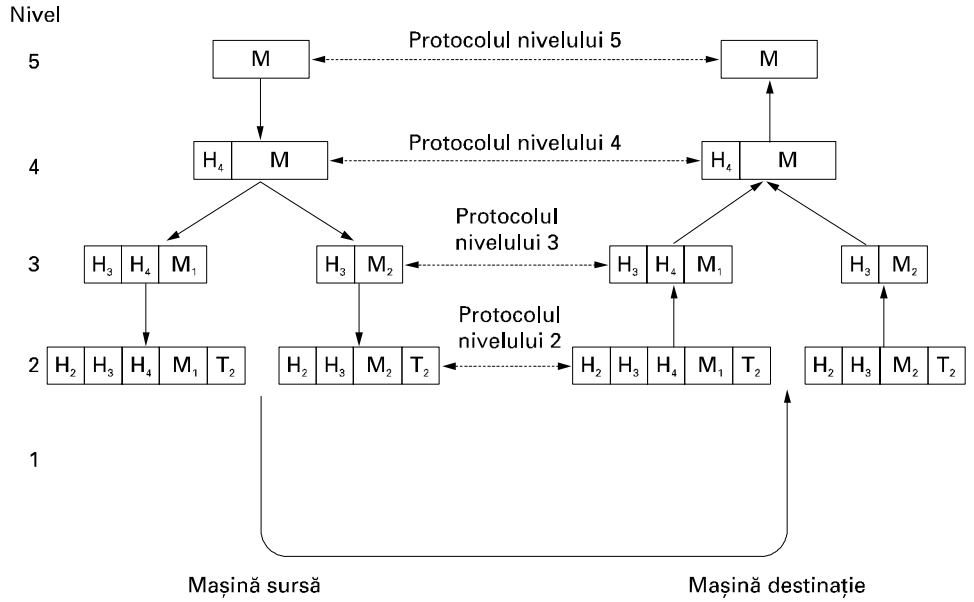


Fig. 1-15. Exemplu de flux de informații pentru suportul comunicării virtuale la nivelul 5.

În continuare, translatorul înmânează mesajul secretarei, care îl trimite, de exemplu, prin fax (protocolul nivelului 1). Când mesajul este primit, el este tradus în franceză și trimis prin interfața 2/3 către filosoful 2. Observați că, atâta timp cât interfețele nu se modifică, fiecare protocol este complet independent de celelalte. Dacă doresc, translatorii pot schimba olandeza cu altă limbă, să spunem finlandeza, cu condiția ca amândoi să se înțeleagă asupra acestui lucru și ca nici unul din ei să nu își modifice interfața cu nivelul 1 sau cu nivelul 3. În mod similar, secretarele pot înlocui faxul cu poșta electronică sau cu telefonul fără a deranja (sau măcar a informa) celelalte niveluri. Fiecare proces poate adăuga anumite informații suplimentare destinate numai procesului său pereche. Aceste informații nu sunt transmise în sus, către nivelul superior.

Să considerăm acum un exemplu mai tehnic: cum se realizează comunicarea la ultimul nivel din rețeaua cu cinci niveluri din fig. 1-15. O aplicație care se execută în nivelul 5 produce un mesaj M și îl furnizează nivelului 4 pentru a-l transmite. Nivelul 4 inserează un **antet** în fața mesajului, pentru a identifica respectivul mesaj și pasează rezultatul nivelului 3. Antetul include informații de control, de exemplu numere de ordine care ajută nivelul 4 de pe mașina de destinație să livreze mesajele în ordinea corectă în cazul în care nivelurile inferioare nu păstrează această ordine. Pe unele niveluri, antetele conțin de asemenea câmpuri de control pentru mărime, timp și alte informații.

În numeroase rețele nu există nici o limită cu privire la mărimea mesajelor transmise în protocolul nivelului 4, dar există aproape întotdeauna o limită impusă de protocolul nivelului 3. În consecință, nivelul 3 trebuie să spargă mesajele primite în unități mai mici, pachete, atașând fiecărui pachet un antet specific nivelului 3. În acest exemplu, M este descompus în două părți, M_1 și M_2.

Nivelul 3 decide ce linie de transmisie să utilizeze și trimite pachetele nivelului 2. Nivelul 2 adaugă nu numai câte un antet pentru fiecare bucată, ci și o încheiere, după care furnizează unitatea rezultantă nivelului 1 pentru a o transmite fizic. În mașina receptoare mesajul este trimis în sus, din

nivel în nivel, pe parcurs fiind eliminate succesiv toate antetele. Nici un antet corespunzător nivelurilor de sub *n* nu este transmis în sus nivelului *n*.

Ceea ce este important de înțeles în fig. 1-15 este relația dintre comunicația virtuală și cea efectivă și diferența între protocoale și interfețe. De exemplu, procesele egale de la nivelul 4 își imaginează conceptual comunicarea ca realizându-se pe „orizontală", utilizând protocolul nivelului 4. Deși fiecare din ele are, probabil, o procedură de genul *TrimiteÎnCealaltăParte* și o alta *PrimeșteDinCealaltăParte*, aceste proceduri nu comunică de fapt cu cealaltă parte, ci cu nivelurile inferioare prin interfața 3/4.

Abstractizarea proceselor pereche este crucială pentru proiectarea întregii rețele. Cu ajutorul ei, această sarcină practic imposibilă poate fi descompusă în probleme de proiectare mai mici, rezolvabile, și anume proiectarea nivelurilor individuale.

Deși Secțiunea 1-3 este intitulată „Programele de rețea", merită să subliniem că nivelurile inferioare dintr-o ierarhie de protocoale sunt implementate frecvent în hardware sau în firmware. Nu e mai puțin adevărat că aici intervin algoritmi complecși, chiar dacă ei sunt înglobați (parțial sau în totalitate) în hardware.

1.3.2 Probleme de proiectare a nivelurilor

O parte din problemele cheie care apar la proiectarea rețelelor de calculatoare sunt prezente în mai multe niveluri. Vom menționa pe scurt unele probleme mai importante.

Fiecare nivel are nevoie de un mecanism pentru a identifica emițătorii și receptorii. Dat fiind că o rețea cuprinde în mod normal numeroase calculatoare, iar o parte dintre acestea dețin mai multe procese, este necesară o modalitate prin care un proces de pe o anumită mașină să specifice cu cine dorește să comunice. Ca o consecință a destinațiilor multiple, pentru a specifica una dintre ele, este necesară o formă de adresare.

Un alt set de decizii de proiectare se referă la regulile pentru transferul de date. În unele sisteme datele circulă într-un singur sens; în altele datele pot circula în ambele sensuri. Protocolul trebuie, de asemenea, să determine câtor canale logice le corespunde conexiunea și care sunt prioritățile acestora. Multe rețele dispun de cel puțin două canale logice pe conexiune, unul pentru date normale și unul pentru date urgente.

Controlul erorilor este o problemă importantă deoarece circuitele fizice de comunicații nu sunt perfecte. Se cunosc multe coduri detectoare și corectoare de erori, dar ambele capete ale conexiunii trebuie să se înțeleagă asupra codului utilizat. În plus, receptorul trebuie să aibă cum să-i spună emițătorului care mesaje au fost primite corect și care nu.

Nu toate canalele de comunicații păstrează ordinea mesajelor trimise. Pentru a putea trata o eventuală pierdere a secvențialității, protocolul trebuie să furnizeze explicit receptorului informația necesară pentru a putea reconstitui mesajul. O soluție evidentă este numerotarea fragmentelor, dar această soluție încă nu rezolvă problema fragmentelor care sosesc la receptor aparent fără legătură cu restul mesajului.

O problemă ce intervine la fiecare nivel se referă la evitarea situației în care un emițător rapid trimite unui receptor lent date la viteză prea mare. Au fost propuse diverse rezolvări și ele vor fi discutate mai târziu. Unele dintre acestea presupun o anumită reacție, directă sau indirectă, prin care receptorul îl informează pe emițător despre starea sa curentă. Altele limitează viteza de transmisie a emițătorului la o valoare stabilită de comun acord cu receptorul. Acest subiect se numește **controlul fluxului**.

O altă problemă care apare la câteva niveluri privește incapacitatea tuturor proceselor de a accepta mesaje de lungime arbitrară. Acest fapt conduce la mecanisme pentru a dezasambla, a transmite și apoi a reasambla mesajele. O problemă asemănătoare apare atunci când procesele insistă să transmită datele în unități atât de mici, încât transmiterea lor separată este ineficientă. În această situație, soluția este să se asambleze împreună mai multe mesaje mici destinate aceluiași receptor și să se dezasambleze la destinație mesajul mare obținut astfel.

Atunci când este neconvenabil sau prea costisitor să se aloce conexiuni separate pentru fiecare pereche de procese comunicante, nivelul implicat în comunicare poate hotărî să utilizeze aceeași conexiune pentru mai multe conversații independente. Atâta timp cât această **mutiplexare** și **demultiplexare** se realizează transparent, ea poate fi utilizată de către orice nivel. Multiplexarea este necesară, de exemplu, în nivelul fizic, unde traficul pentru toate conexiunile trebuie să fie transmis prin cel mult câteva circuite fizice.

Atunci când există mai multe căi între sursă și destinație, trebuie ales un anumit drum. Uneori această decizie trebuie împărțită pe două sau mai multe niveluri. De exemplu, este posibil ca trimiterea unor date de la Londra la Roma să necesite atât o decizie la nivel înalt pentru alegerea ca țară de tranzit a Franței sau a Germaniei - în funcție de legile lor de protejare a secretului datelor - cât și o decizie de nivel scăzut pentru alegerea unuia din multele trasee posibile, pe baza traficului curent. Acest subiect poartă numele de **dirijare** sau **rutare** (**routing**).

1.3.3 Servicii orientate pe conexiuni și servicii fără conexiuni

Nivelurile pot oferi nivelurilor de deasupra lor două tipuri de servicii: orientate pe conexiuni și fără conexiuni. În această secțiune vom arunca o privire asupra acestor două tipuri și vom examina diferențele între ele.

Serviciul orientat pe conexiuni este modelat pe baza sistemului telefonic. Când vrei să vorbești cu cineva, mai întâi ridici receptorul, apoi formezi numărul, vorbești și închizi. Similar, pentru a utiliza un serviciu orientat pe conexiuni, beneficiarul trebuie mai întâi să stabilească o conexiune, să folosească această conexiune și apoi să o elibereze. În esență conexiunea funcționează ca o țeavă: emițătorul introduce obiectele (biții) la un capăt, iar receptorul le scoate afară, în aceeași ordine, la celălalt capăt. În majoritatea cazurilor ordinea este menținută, astfel încât biții să ajungă în aceeași ordine în care au fost trimiși.

În anumite cazuri când se stabilește o conexiune, transmițătorul, receptorul și subrețeaua negociază parametrii care vor fi folosiți, cum sunt dimensiunea maximă a mesajului, calitatea impusă a serviciilor, și alte probleme de acest tip. De obicei, una dintre părți face o propunere și cealaltă parte poate să o accepte, să o rejecteze sau să facă o contrapropunere.

Serviciul fără conexiuni este modelat pe baza sistemului poștal. Toate mesajele (scrisorile) conțin adresele complete de destinație și fiecare mesaj circulă în sistem independent de celelalte. În mod normal, atunci când două mesaje sunt trimise la aceeași destinație, primul expediat este primul care ajunge. Totuși, este posibil ca cel care a fost expediat primul să întârzie și să ajungă mai repede al doilea. În cazul unui serviciu orientat pe conexiuni, așa ceva este imposibil.

Fiecare serviciu poate fi caracterizat printr-o **calitate a serviciului**. Unele servicii sunt sigure în sensul că nu pierd date niciodată. De obicei, un serviciu sigur se implementează obligând receptorul să confirme primirea fiecărui mesaj, astfel încât expeditorul să fie sigur că mesajul a ajuns la destinație. Procesul de confirmare introduce un timp suplimentar și întârzieri. Aceste dezavantaje sunt adesea acceptate, însă uneori ele trebuie evitate.

Transferul de fișiere este una din situațiile tipice în care este adecvat un serviciu sigur orientat pe conexiuni. Proprietarul fișierului dorește să fie sigur că toți biții ajung corect și în aceeași ordine în care au fost trimiși. Foarte puțini utilizatori ai transferului de fișiere ar prefera un serviciu care uneori amestecă sau pierde câțiva biți, chiar dacă acest serviciu ar fi mult mai rapid.

Serviciul sigur orientat pe conexiuni admite două variante: secvențele de mesaje și fluxurile de octeți. Prima variantă menține delimitarea între mesaje. Când sunt trimise două mesaje de 1024 de octeți, ele vor sosi sub forma a două mesaje distincte de 1024 de octeți, niciodată ca un singur mesaj de 2048 de octeți. În a doua variantă, conexiunea este un simplu flux de octeți și nu există delimitări între mesaje. Când receptorul primește 2048 de octeți, nu există nici o modalitate de a spune dacă ei au fost trimiși sub forma unui mesaj de 2048 octeți, a două mesaje de 1024 de octeți sau a 2048 mesaje de câte 1 octet. Dacă paginile unei cărți sunt expediate unei mașini fotografice de tipărit printr-o rețea, sub formă de mesaje, atunci delimitarea mesajelor poate fi importantă. Pe de altă parte, în cazul unui utilizator care se conectează la un server aflat la distanță, este nevoie numai de un flux de octeți de la calculatorul utilizatorului la server. Delimitarea mesajelor nu mai este relevantă.

Așa cum am menționat mai sus, întârzierile introduse de confirmări sunt inacceptabile pentru unele aplicații. O astfel de aplicație se referă la traficul de voce digitizată. Pentru abonații telefonici este preferabil să existe puțin zgomot pe linie sau să audă ocazional câte un cuvânt distorsionat decât să se producă o întârziere din cauza așteptării confirmării. Similar, atunci când se transmite o videoconferință, câțiva pixeli diferiți nu reprezintă o problemă, în schimb întreruperile pentru a corecta erorile ar fi extrem de supărătoare.

Nu orice aplicație necesită conexiuni. De exemplu, în măsura în care poșta electronică devine ceva tot mai uzual, se poate să nu apară foarte curând publicitatea prin poștă electronică? Expeditorul de publicitate prin poștă electronică probabil că nu vrea să se complice stabilind și apoi eliberând o conexiune doar pentru un singur mesaj. Nici furnizarea la destinație cu o rată de corectitudine de 100% nu este esențială, mai ales dacă lucrul acesta costă mai mult. Tot ceea ce se cere este un mijloc de a trimite un singur mesaj cu o probabilitate mare de a ajunge la destinație, dar fără o garanție în acest sens. Serviciul nesigur (adică neconfirmat) fără conexiuni este deseori numit **serviciu datagramă**, prin analogie cu serviciul de telegrame - care, la rândul său, nu prevede trimiterea unei confirmări către expeditor.

În alte situații, avantajul de a nu fi necesară stabilirea unei conexiuni pentru a trimite un mesaj scurt este de dorit, dar siguranța este de asemenea esențială. Aceste aplicații pot utiliza **serviciul datagramă confirmat**. Este ca și cum ai trimite o scrisoare recomandată și ai solicita o confirmare de primire. În clipa în care sosește confirmarea, expeditorul este absolut sigur că scrisoarea a fost livrată la destinația corectă și nu a fost pierdută pe drum.

Mai există un serviciu, și anume **serviciul cerere-răspuns**. În acest serviciu emițătorul transmite o singură datagramă care conține o cerere; replica primită de la receptor conține răspunsul. În această categorie intră, de exemplu, un mesaj către biblioteca locală în care se întreabă unde este vorbită limba Uighur. Serviciul cerere-răspuns este utilizat în mod frecvent pentru a implementa comunicarea în modelul client-server: clientul lansează o cerere și serverul răspunde la ea. În fig. 1-16 sunt rezumate tipurile de servicii discutate mai sus.

Conceptul de a utiliza comunicații nesigure poate părea derutant la început. La urma urmei, de ce ar prefera cineva comunicațiile nesigure în locul comunicațiilor sigure? Mai întâi, comunicațiile sigure (ceea ce înseamnă, pentru noi, confirmate) pot să nu fie disponibile. De exemplu, Ethernet-ul nu oferă comunicații sigure. Pachetele pot fi uneori alterate în timpul tranzitului. Urmează ca protocoalele nivelurilor superioare să se ocupe de această problemă.

	Serviciu	Exemplu
Orientate pe conexiuni	Flux de mesaje sigur	Secvență de pagini
	Flux de octeți sigur	Conectare la distanță
	Conexiune nesigură	Voce digitizată
Fără conexiuni	Datagramă nesigură	Publicitate prin e-mail
	Datagramă confirmată	Scrisori cu confirmare
	Cerere-răspuns	Interogări baze de date

Fig. 1-16. Șase tipuri diferite de servicii.

În al doilea rând, întârzierile inerente în cazul în care se oferă servicii sigure ar putea fi inacceptabile, mai ales în cazul aplicațiilor de timp real cum sunt aplicațiile multimedia. Pentru aceste motive, comunicațiile sigure cât și cele nesigure coexistă.

1.3.4 Primitive de serviciu

Un serviciu este specificat formal printr-un set de **primitive** (operații) puse la dispoziția utilizatorului care folosește serviciul. Aceste primitive comandă serviciului să execute anumite acțiuni sau să raporteze despre acțiunile executate de o entitate pereche. Dacă stiva de protocoale este localizată în sistemul de operare, așa cum se întâmplă de cele mai multe ori, primitivele sunt în mod normal apeluri sistem. Aceste apeluri cauzează o trecere a sistemului de operare în modul nucleu (kernel), care preia controlul mașinii pentru a trimite pachetele necesare.

Setul de primitive disponibile depinde de natura serviciului oferit. Primitivele serviciilor orientate pe conexiuni sunt diferite de cele ale serviciilor fără conexiuni. Ca un exemplu minimal de primitive de serviciu care pot fi oferite pentru a implementa un flux de octeți într-un mediu client-server, putem considera primitivele listate în fig. 1-17.

Primitiva	Semnificația
LISTEN (Ascultă)	Blocare în așteptarea unei conexiuni
CONNECT (Conectează)	Stabilirea unei conexiuni cu o entitate pereche aflată în așteptare
RECEIVE (Primește)	Blocare în așteptarea unui mesaj
SEND (Trimite)	Trimite un mesaj entității pereche
DISCONNECT (Deconectează)	Termină o conexiune

Fig. 1-17. Cinci primitive de serviciu pentru implementarea unui serviciu simplu orientat pe conexiune.

Aceste primitive pot fi folosite în următorul mod: mai întâi serverul execută LISTEN pentru a indica faptul că este pregătit să accepte conexiuni. Un mod obișnuit de a implementa LISTEN este a

face un apel de sistem blocant. După execuția primitivei, procesul server este blocat până la apariția unei cereri de conectare.

Apoi procesul client execută CONNECT pentru a stabili o conexiune cu serverul. Apelul CONNECT trebuie să specifice cu cine se dorește conectarea, așa că ar putea avea un parametru prin care se transmite adresa serverului. De cele mai multe ori, sistemul de operare va trimite un prim pachet entității pereche cerându-i să se conecteze, după cum este arătat de (1) în fig. 1-18. Procesul client este suspendat până când apare un răspuns. Când pachetul ajunge la server, el este procesat de sistemul de operare al acestuia. Când sistemul de operare observă că pachetul cere o conexiune, verifică dacă există vreun ascultător. Dacă da, va face doua lucruri: va debloca ascultătorul și va trimite înapoi o confirmare (2). Sosirea acestei confirmări eliberează apoi clientul. În acest moment, atât clientul cât și serverul sunt în execuție și au stabilit o conexiune între ei. Este important de observat că secvența de confirmare (2) este generată de codul protocolului însuși, nu ca răspuns al unei primitive de la nivelul utilizatorului. Dacă apare o cerere de conexiune și nu există nici un ascultător, rezultatul este nedefinit. În anumite sisteme, pachetul poate fi păstrat un scurt timp într-o coadă, anticipând o eventuală comandă LISTEN.

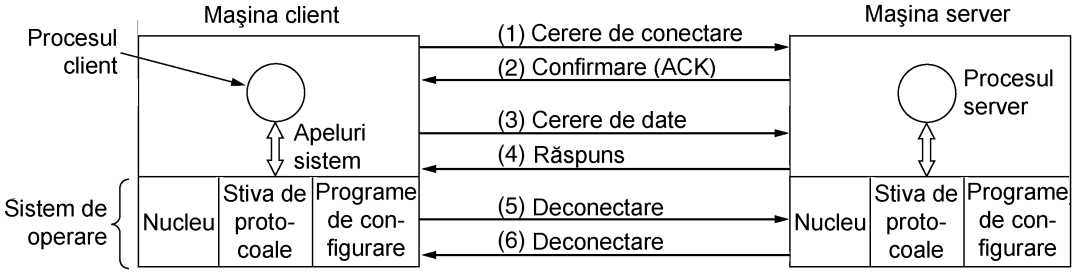


Fig. 1-18. Pachetele trimise într-o simplă interacțiune client-server pe o rețea orientată pe conexiuni.

Analogia evidentă între acest protocol și viața reală este cazul clientului care sună la directorul departamentului de service al unei companii. Directorul stă lângă telefon pentru a putea răspunde în cazul în care acesta sună. Clientul face un apel. Când directorul ridică receptorul, conexiunea este stabilită.

Pasul următor este ca serverul să execute RECEIVE pentru a se pregăti să accepte prima cerere. În mod normal serverul face această operație de îndată ce a fost eliberat din blocarea impusă de LISTEN, înainte să ajungă confirmarea înapoi la client. Apelul RECEIVE blochează serverul.

Apoi clientul execută SEND pentru a transmite cererea sa (3) urmat de execuția unui RECEIVE pentru a obține răspunsul.

Sosirea pachetului de cerere la mașina server deblochează procesul server astfel încât acesta să poată procesa cererea. După ce a terminat lucrul, folosește SEND pentru a răspunde clientului (4). Sosirea acestui pachet deblochează clientul care poate acum să analizeze răspunsul obținut. Daca mai există cereri din partea clientului, acesta le poate face acum. Dacă a terminat, poate folosi DISCONNECT pentru a termina conexiunea. De obicei, apelul inițial DISCONNECT este blocant, suspendând clientul și trimițând un pachet către server pentru a-i comunica faptul ca respectiva conexiune nu mai este necesară (5). Când serverul primește pachetul, el lansează un DISCONNECT propriu, confirmând cererea clientului și eliberând conexiunea. Când pachetul serverului (6) ajunge

înapoi la mașina clientului, procesul client este eliberat și conexiunea este întreruptă. Foarte pe scurt, așa funcționează comunicațiile orientate pe conexiuni.

Desigur, viața nu este simplă. Multe dintre lucruri pot să nu funcționeze corect. Sincronizarea poate fi proastă (de exemplu, daca se încearcă un CONNECT înainte de LISTEN), pachetele se pot pierde și multe altele. Vom studia toate acestea în detaliu ceva mai târziu, dar deocamdată fig. 1-18 rezumă pe scurt modul în care ar putea să funcționeze o comunicație client-server într-o rețea orientată pe conexiuni.

Știind că acele șase pachete sunt necesare pentru a realiza acest protocol, cititorul se poate întreba de ce nu se folosește un protocol fără conexiune în locul său. Răspunsul este că ar fi posibil într-o lume perfectă, și atunci ar fi nevoie de numai două pachete: unul pentru cerere și unul pentru răspuns. Oricum, în cazul real cu mesaje lungi în oricare dintre direcții (de exemplu un fișier de 1 MB), cu erori de transmisie și cu pachete pierdute, situația se modifică. Dacă răspunsul ar avea sute de pachete, dintre care unele s-ar putea pierde în timpul transmisiei, cum ar putea clientul să își dea seama că unele piese lipsesc? Cum ar putea ști clientul dacă ultimul pachet recepționat este de fapt ultimul pachet trimis? Să presupunem că de la client se face o cerere pentru un al doilea fișier. Cum ar putea clientul să diferențieze pachetele din cel de-al doilea fișier de eventualele pachete pierdute din primul fișier? Pe scurt, în lumea reală, un simplu protocol cerere-răspuns implementat într-o rețea nesigură este de cele mai multe ori inadecvat. În cap. 3 vom studia în detaliu o largă varietate de protocoale, care pot rezolva aceste probleme și altele similare. Pentru moment însă este de ajuns să spunem că a avea un flux de octeți sigur și ordonat între procese este de multe ori foarte convenabil.

1.3.5 Relația dintre servicii și protocoale

Deși sunt adesea confundate, serviciile și protocoalele reprezintă concepte distincte. Diferența între ele este atât de importantă, încât o subliniem din nou în această secțiune. Un *serviciu* este un set de primitive (operații) pe care un nivel le furnizează nivelului de deasupra sa. Serviciul definește ce operații este pregătit nivelul să realizeze pentru utilizatorii săi, dar nu spune nimic despre cum sunt implementate aceste operații. Un serviciu este definit în contextul unei interfețe între două niveluri, nivelul inferior fiind furnizorul serviciului și nivelul superior fiind utilizatorul serviciului.

Prin contrast, un *protocol* este un set de reguli care guvernează formatul și semnificația cadrelor, pachetelor sau mesajelor schimbate între ele de entitățile pereche dintr-un nivel. Entitățile folosesc protocoale pentru a implementa definițiile serviciului lor. Ele sunt libere să își schimbe protocoalele după cum doresc, cu condiția să nu modifice serviciul pe care îl văd utilizatorii. În acest fel, serviciul și protocolul sunt complet decuplate.

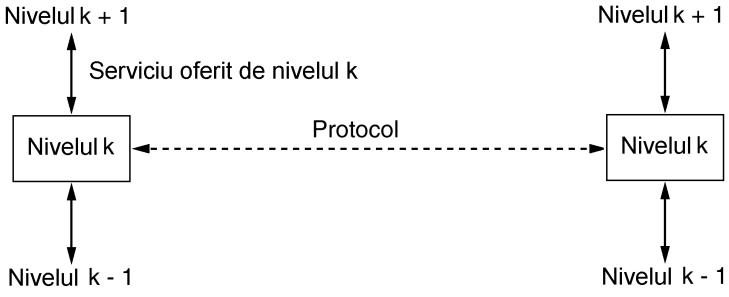


Fig. 1-19. Relația dintre un server și un protocol.

Cu alte cuvinte, serviciile sunt legate de interfețele dintre niveluri, după cum este ilustrat și în fig. 1-19. Prin contrast, protocoalele sunt legate de pachetele trimise între entitățile pereche de pe diferite mașini. Este important să nu existe confuzii între cele două concepte.

Merită să facem o analogie cu limbajele de programare. Un serviciu este ca un tip de date abstracte sau ca un obiect într-un limbaj orientat pe obiecte. Acesta definește operațiile care pot fi aplicate pe un obiect, dar nu specifică modul de implementare a operațiilor. Un protocol se referă la *implementarea* serviciului și nu este vizibil pentru utilizatorul serviciului.

Multe protocoale mai vechi nu făceau diferența între serviciu și protocol. Ca urmare, un nivel tipic putea avea o primitivă de serviciu SEND PACKET în care utilizatorul furniza o referință către un pachet complet asamblat. Acest aranjament însemna că toate modificările protocolului erau imediat vizibile pentru utilizatori. Majoritatea proiectanților de rețele privesc acum un astfel de mecanism ca pe o eroare gravă.

1.4 MODELE DE REFERINȚĂ

Acum, după ce am discutat la modul abstract structura pe niveluri a rețelelor, a sosit timpul să studiem câteva exemple. În următoarele două secțiuni vom discuta două arhitecturi de rețea importante, modelul de referință OSI și modelul de referință TCP/IP. Deși *protocoalele* asociate cu modelul OSI nu sunt folosite aproape deloc, *modelul* în sine este destul de general și încă valabil, iar caracteristicile puse în discuție la fiecare nivel sunt în continuare foarte importante. Modelul TCP/IP are caracteristici opuse: modelul în sine nu este foarte util, dar protocoalele sunt folosite pe scară largă. Din acest motiv, le vom studia pe fiecare în detaliu. În plus, uneori poți învăța mai multe din eșecuri decât din succese.

1.4.1 Modelul de referință OSI

Modelul OSI este prezentat în fig. 1-16 (mai puțin mediul fizic). Acest model se bazează pe o propunere dezvoltată de către Organizația Internațională de Standardizare (International Standards Organization - ISO) ca un prim pas către standardizarea internațională a protocoalelor folosite pe diferite niveluri (Day și Zimmermann, 1983). A fost revizuit în 1995 (Day, 1995). Modelul se numește **ISO OSI** (**Open Systems Interconnection**, rom: interconectarea sistemelor deschise), pentru că el se ocupă de conectarea sistemelor deschise - adică de sisteme deschise comunicării cu alte sisteme. În continuare vom folosi mai ales termenul prescurtat de model OSI.

Modelul OSI cuprinde șapte niveluri. Principiile aplicate pentru a se ajunge la cele șapte niveluri sunt următoarele:

1. Un nivel trebuie creat atunci când este nevoie de un nivel de abstractizare diferit.
2. Fiecare nivel trebuie să îndeplinească un rol bine definit.
3. Funcția fiecărui nivel trebuie aleasă acordându-se atenție definirii de protocoale standardizate pe plan internațional.
4. Delimitarea nivelurilor trebuie făcută astfel încât să se minimizeze fluxul de informații prin interfețe.

5. Numărul de niveluri trebuie să fie suficient de mare pentru a nu fi nevoie să se introducă în același nivel funcții diferite și suficient de mic pentru ca arhitectura să rămână funcțională.

În continuare vom discuta fiecare nivel al modelului, începând cu nivelul cel mai de jos. Modelul OSI nu reprezintă în sine o arhitectură de rețea, pentru că nu specifică serviciile și protocoalele utilizate la fiecare nivel. Modelul spune numai ceea ce ar trebui să facă fiecare nivel. ISO a produs de asemenea standarde pentru fiecare nivel, însă aceste standarde nu fac parte din modelul de referință propriu-zis. Fiecare din standardele respective a fost publicat ca un standard internațional separat.

Nivelul fizic

Nivelul fizic se ocupă de transmiterea biților printr-un canal de comunicație. Proiectarea trebuie să garanteze că atunci când unul din capete trimite un bit 1, acesta e receptat în cealaltă parte ca un bit 1, nu ca un bit 0. Problemele tipice se referă la câți volți trebuie utilizați pentru a reprezenta un 1 și câți pentru un 0, dacă transmisia poate avea loc simultan în ambele sensuri, cum este stabilită conexiunea inițială și cum este întreruptă când au terminat de comunicat ambele părți, câți pini are conectorul de rețea și la ce folosește fiecare pin. Aceste aspecte de proiectare au o legătură strânsă cu interfețele mecanice, electrice, funcționale și procedurale, ca și cu mediul de transmisie situat sub nivelul fizic.

Nivelul legătură de date

Sarcina principală a **nivelului legăturii de date** este de a transforma un mijloc oarecare de transmisie într-o linie care să fie disponibilă nivelului rețea fără erori de transmisie nedetectate. Nivelul legătură de date realizează această sarcină obligând emițătorul să descompună datele de intrare în **cadre de date** (în mod tipic, câteva sute sau câteva mii de octeți) și să transmită cadrele secvențial. Dacă serviciul este sigur, receptorul confirmă fiecare cadru trimițând înapoi un cadru de confirmare pozitivă.

O altă problemă care apare la nivelul legătură de date (și, de asemenea, la majoritatea nivelurilor superioare) este evitarea inundării unui receptor lent cu date provenite de la un emițător rapid. În acest scop sunt necesare mecanisme de reglare a traficului care să permită emițătorului să afle cât spațiu tampon deține receptorul la momentul curent. Controlul traficului și tratarea erorilor sunt deseori integrate. Rețelele cu difuzare determină în nivelul legătură de date o problemă suplimentară: cum să fie controlat accesul la canalul partajat. De această problemă se ocupă un subnivel special al nivelului legătură de date și anume subnivelul de control al accesului la mediu.

Nivelul rețea

Nivelul rețea se ocupă de controlul funcționării subrețelei. O problemă cheie în proiectare este determinarea modului în care pachetele sunt dirijate de la sursă la destinație. Dirijarea se poate baza pe tabele statistice care sunt „cablate” intern în rețea și care sunt schimbate rar. Traseele pot fi de asemenea stabilite la începutul fiecărei conversații, de exemplu la începutul unei sesiuni la terminal (de ex. o operație de login pe o mașină la distanță). În sfârșit, dirijarea poate fi foarte dinamică, traseele determinându-se pentru fiecare pachet în concordanță cu traficul curent din rețea.

Dacă în subrețea există prea multe pachete simultan, ele vor intra unul pe traseul celuilalt și astfel se vor produce gâtuiri. Controlul unor astfel de congestii îi revine tot nivelului rețea. Mai general, calitatea serviciilor oferite (întârziere, timp de tranzitare, fluctuații, etc.) este tot o responsabilitate a nivelului rețea.

Multe probleme pot apărea când un pachet trebuie să călătorească dintr-o rețea în alta ca să ajungă la destinație. Modul de adresare folosit de a doua rețea poate să difere de cel pentru prima.

A doua rețea poate chiar să nu accepte deloc pachetul pentru că este prea mare. De asemenea, protocoalele pot fi diferite și așa mai departe. Rezolvarea acestor probleme în vederea interconectării rețelelor eterogene este sarcina nivelului rețea. În rețelele cu difuzare, problema dirijării este simplă, astfel că nivelul rețea este deseori subțire sau chiar nu există deloc.

Nivelul transport

Rolul principal al nivelului transport este să accepte date de la nivelul sesiune, să le descompună, dacă este cazul, în unități mai mici, să transfere aceste unități nivelului rețea și să se asigure că toate fragmentele sosesc corect la celălalt capăt. În plus, toate acestea trebuie făcute eficient și într-un mod care izolează nivelurile de mai sus de inevitabilele modificări în tehnologia echipamentelor.

Nivelul transport determină, de asemenea, ce tip de serviciu să furnizeze nivelului sesiune și, în final, utilizatorilor rețelei. Cel mai obișnuit tip de conexiune transport este un canal punct-la-punct fără erori care furnizează mesajele sau octeții în ordinea în care au fost trimiși. Alte tipuri posibile de servicii de transport sunt transportul mesajelor individuale - fără nici o garanție în privința ordinii de livrare - și difuzarea mesajelor către destinații multiple. Tipul serviciului se determină când se stabilește conexiunea. (Ca un comentariu secundar: este imposibil de obținut un canal fără erori; ceea ce oamenii înțeleg prin această expresie este că rata erorilor este destul de mică pentru a fi ignorată în practică).

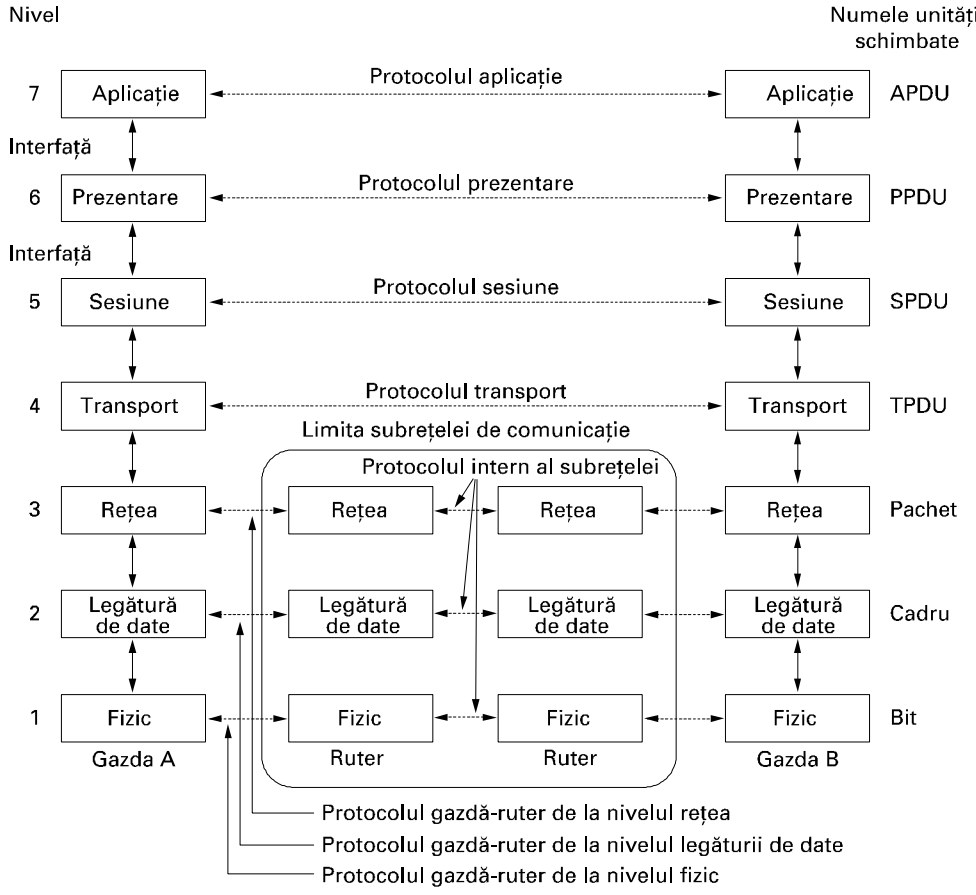


Fig. 1-20. Modelul de referință OSI.

Nivelul transport este un adevărat nivel capăt-la-capăt, de la sursă la destinație. Cu alte cuvinte, un program de pe mașina sursă poartă o conversație cu un program similar de pe mașina destinație, folosind în acest scop antetele mesajelor și mesaje de control. În nivelurile inferioare protocoalele au loc între fiecare mașină și vecinii săi imediați (niveluri înlănțuite), și nu direct între mașinile sursă și destinație (niveluri capăt-la-capăt), care pot fi separate de numeroase rutere. Diferența între nivelurile de la 1 până la 3, care sunt înlănțuite și nivelurile de la 4 la 7, care sunt capăt-la-capăt, este ilustrată în fig. 1-20.

Nivelul sesiune

Nivelul sesiune permite utilizatorilor de pe mașini diferite să stabilească între ei sesiuni. Sesiunile oferă diverse servicii, incluzând controlul dialogului (respectarea ordinii în raport cu dreptul de a transmite), **gestionarea jetonului** (prevenirea situației în care două entități încearcă aceeași operație critică în același timp) și **sincronizarea** (introducerea de puncte de control pe parcursul transmisiilor lungi, astfel încât, în cazul unui eșec, acestea să poată fi reluate de unde rămăseseră).

Nivelul prezentare

În particular, spre deosebire de nivelurile inferioare, care se ocupă numai de transferul biților dintr-un loc în altul, nivelul prezentare se ocupă de sintaxa și semantica informațiilor transmise. Pentru a face posibilă comunicarea între calculatoare cu reprezentări diferite ale datelor, structurile de date care se schimba între ele pot fi definite într-un mod abstract, alături de o codificare standardizată ce va fi utilizată „pe cablu”. Nivelul prezentare gestionează aceste structuri de date abstracte și permite definirea și comunicarea unor structuri de date de nivel mai înalt (de ex. înregistrări bancare).

Nivelul aplicație

Nivelul aplicație conține o varietate de protocoale frecvent utilizate. Un exemplu de protocol utilizat pe scară largă este **HTTP** (**HyperText Transfer Protocol**, rom: protocol de transfer al hipertextului), care sta la baza **WWW** (**World Wide Web**, rom: rețea de întindere planetară). Atunci când un program de navigare (browser) accesează o pagină Web, el trimite serverului numele paginii pe care o dorește folosind HTTP. Serverul va trimite ca răspuns pagina. Alte protocoale de aplicație sunt folosite pentru transferul fișierelor, poștă electronica, știri în rețea.

1.4.2 Modelul de referință TCP/IP

Să ne îndreptăm acum atenția de la modelul de referință OSI spre modelul de referință utilizat de strămoșul tuturor rețelelor de calculatoare, ARPANET-ul, și de succesorul său, Internet-ul. Deși vom prezenta mai târziu o scurtă istorie a ARPANET-ului, este util să menționăm acum câteva aspecte esențiale. ARPANET a fost o rețea de cercetare sponsorizată de către DoD (U.S. Department of Defense, rom: Departamentul de Apărare al Statelor Unite). În cele din urmă, rețeaua a ajuns să conecteze între ele, utilizând linii telefonice închiriate, sute de rețele universitare și guvernamentale. Atunci când au fost adăugate, mai târziu, rețele prin satelit și radio, interconectarea acestora cu protocoalele existente a pus diferite probleme. Era nevoie de o nouă arhitectură de referință. De aceea, posibilitatea de a interconecta fără probleme mai multe tipuri de rețele a reprezentat de la bun început un obiectiv de proiectare major. Această arhitectură a devenit cunoscută mai târziu sub denumirea de **modelul de referință TCP/IP**, dată după numele celor două protocoale fundamentale utilizate. Arhitectura respectivă a fost definită prima dată în (Cerf și Kahn, 1974). O perspectivă ul-

terioară este prezentată în (Leiner ș.a., 1985). Filosofia de proiectare din spatele modelului este discutată în (Clark, 1988).

Dată fiind îngrijorarea Departamentului de Apărare că o parte din prețioasele sale gazde, rutere și porți de interconectare ar putea fi distruse dintr-un moment în altul, un alt obiectiv major a fost ca rețeaua să poată supraviețui pierderii echipamentelor din subrețea fără a fi întrerupte conversațiile existente. Cu alte cuvinte, DoD dorea ca, atâta timp cât funcționau mașina sursă și mașina destinație, conexiunile să rămână intacte, chiar dacă o parte din mașini sau din liniile de transmisie erau brusc scoase din funcțiune. Mai mult, era nevoie de o arhitectură flexibilă, deoarece se aveau în vedere aplicații cu cerințe divergente, mergând de la transferul de fișiere până la transmiterea vorbirii în timp real.

Nivelul internet

Toate aceste cerințe au condus la alegerea unei rețele cu comutare de pachete bazată pe un nivel inter-rețea fără conexiuni. Acest nivel, numit **nivelul internet**, este axul pe care se centrează întreaga arhitectură. Rolul său este de a permite gazdelor să emită pachete în orice rețea și a face ca pachetele să circule independent până la destinație (fiind posibil ca aceasta să se găsească pe o altă rețea). Pachetele pot chiar să sosească într-o ordine diferită față de cea în care au fost trimise, caz în care – dacă se dorește livrarea lor ordonată - rearanjarea cade în sarcina nivelurilor superioare. De observat că „internet" este folosit aici într-un sens generic, chiar dacă acest nivel este prezent și în Internet.

Aici, analogia este cu sistemul de poștă (clasică). O persoană dintr-o anumită țară poate depune într-o cutie poștală mai multe scrisori internaționale și, cu puțin noroc, majoritatea scrisorilor vor ajunge la adresa corectă din țara de destinație. Probabil că scrisorile vor trece pe drum prin mai multe oficii de cartare, dar acest lucru se face transparent pentru utilizatori. Mai mult, faptul că fiecare țară (adică fiecare rețea) are propriile timbre, propriile mărimi favorite de plicuri și propriile reguli de livrare este ascuns beneficiarilor.

Nivelul internet definește oficial un format de pachet și un protocol numit **IP** (**Internet Protocol**, rom: protocol Internet). Sarcina nivelului internet este să livreze pachete IP către destinație. Problemele majore se referă la dirijarea pachetelor și evitarea congestiei. În consecință, este rezonabil să spunem că nivelul internet din TCP/IP funcționează asemănător cu nivelul rețea din OSI. Fig. 1-21 arată această corespondență.

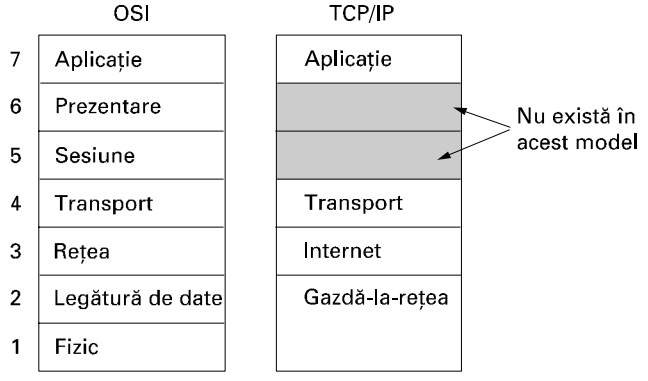


Fig. 1-21. Modelul de referință TCP/IP.

Nivelul transport

Nivelul situat deasupra nivelului internet din modelul TCP/IP este frecvent numit **nivelul transport**. Acesta este proiectat astfel, încât să permită conversații între entitățile pereche din gazdele sursă și, respectiv, destinație, la fel ca în nivelul transport OSI. În acest sens au fost definite două protocoale capăt-la-capăt. Primul din ele, **TCP** (**Transmission Control Protocol**, rom: protocolul de control al transmisiei), este un protocol sigur orientat pe conexiuni care permite ca un flux de octeți trimiși de pe o mașină să ajungă fără erori pe orice altă mașină din inter-rețea. Acest protocol fragmentează fluxul de octeți în mesaje discrete și pasează fiecare mesaj nivelului internet. La destinație, procesul TCP receptor reasamblează mesajele primite într-un flux de ieșire. TCP tratează totodată controlul fluxului pentru a se asigura că un emițător rapid nu inundă un receptor lent cu mai multe mesaje decât poate acesta să prelucreze.

Al doilea protocol din acest nivel, **UDP** (**User Datagram Protocol**, rom: protocolul datagramelor utilizator), este un protocol nesigur, fără conexiuni, destinat aplicațiilor care doresc să utilizeze propria lor secvențiere și control al fluxului, și nu pe cele asigurate de TCP. Protocolul UDP este de asemenea mult folosit pentru interogări rapide întrebare-răspuns, client-server și pentru aplicații în care comunicarea promptă este mai importantă decât comunicarea cu acuratețe, așa cum sunt aplicațiile de transmisie a vorbirii și a imaginilor video. Relația dintre IP, TCP și UDP este prezentată în fig. 1-22. De când a fost dezvoltat acest model, IP a fost implementat pe multe alte rețele.

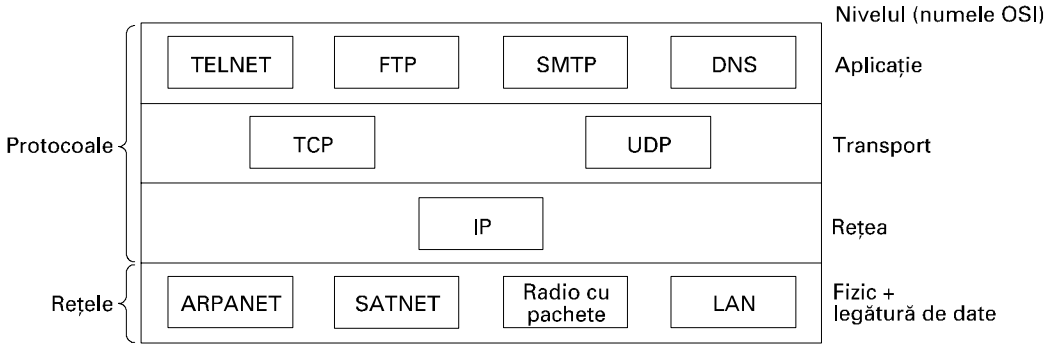


Fig. 1-22. Protocoale și rețele din modelul TCP/IP inițial.

Nivelul aplicație

Modelul TCP/IP nu conține niveluri sesiune sau prezentare. Acestea nu au fost incluse pentru că nu s-a simțit nevoia lor. Experiența modelului OSI a dovedit că această viziune a fost corectă: în majoritatea aplicațiilor, nivelurile respective nu sunt de mare folos.

Deasupra nivelului transport se află **nivelul aplicație**. Acesta conține toate protocoalele de nivel mai înalt. Așa cum se vede din fig. 1-22, primele protocoale de acest gen includeau terminalul virtual (TELNET), transferul de fișiere (FTP) și poșta electronică (SMTP). Protocolul de terminal virtual permite unui utilizator de pe o mașină să se conecteze și să lucreze pe o mașină aflată la distanță. Protocolul de transfer de fișiere pune la dispoziție o modalitate de a muta eficient date de pe o mașină pe alta. Poșta electronică a fost la origine doar un tip de transfer de fișiere, dar ulterior a fost dezvoltat un protocol specializat (SMTP – Simple Mail Transfer Protocol, rom: Protocol simplu de transfer al poștei) pentru acest serviciu. Pe parcursul anilor, la aceste protocoale s-au adăugat multe altele, așa cum sunt Serviciul Numelor de Domenii (Domain Name Service - DNS) pentru stabilirea corespondenței dintre numele gazdelor și adresele rețelelor, NNTP, protocolul

utilizat pentru a transfera articole de știri USENET, HTTP, folosit pentru aducerea paginilor de pe Web și multe altele.

Nivelul gazdă-rețea

Sub nivelul internet se află necunoscutul. Modelul de referință TCP/IP nu spune mare lucru despre ce se întâmplă acolo, însă menționează că gazda trebuie să se lege la rețea, pentru a putea trimite pachete IP, folosind un anumit protocol. Acest protocol nu este definit și variază de la gazdă la gazdă și de la rețea la rețea. Cărțile și articolele despre TCP/IP rareori discută despre acest protocol.

1.4.3 O comparație între modelele de referință OSI și TCP

Modelele de referință OSI și TCP/IP au multe lucruri în comun. Amândouă se bazează pe conceptul unei stive de protocoale independente. De asemenea, funcționalitatea nivelurilor este în linii mari similară. De exemplu, în ambele modele, nivelurile până la nivelul transport inclusiv sunt necesare pentru a pune la dispoziția proceselor care doresc să comunice un serviciu de transport capăt-la-capăt independent de rețea. Nivelurile respective formează furnizorul de transport. Din nou, în ambele modele, nivelurile de deasupra transportului sunt beneficiari orientați pe aplicații ai serviciului de transport.

În pofida acestor similitudini fundamentale, între cele două modele există și multe deosebiri. În această secțiune ne vom concentra asupra diferențelor cheie dintre cele două modele de referință. Este important de subliniat că vom compara aici *modelele de referință*, nu *stivele de protocoale* corespunzătoare. Protocoalele propriu-zise vor fi discutate mai târziu. Pentru o întreagă carte consacrată comparației și diferențelor dintre TCP/IP și OSI, a se vedea (Piscitello și Chapin, 1993).

Trei concepte sunt esențiale pentru modelul OSI:

1. Servicii
2. Interfețe
3. Protocoale

Probabil că cea mai mare contribuție a modelului OSI este că a făcut explicită diferența între aceste trei concepte. Fiecare nivel realizează niște servicii pentru nivelul situat deasupra sa. Definiția *serviciului* spune ce face nivelul, nu cum îl folosesc entitățile de deasupra sa sau cum funcționează nivelul. El definește semantica nivelului.

Interfața unui nivel spune proceselor aflate deasupra sa cum să facă accesul. Interfața precizează ce reprezintă parametrii și ce rezultat se obține. Nici interfața nu spune nimic despre funcționarea internă a nivelului.

În sfârșit, *protocoalele* pereche folosite într-un nivel reprezintă treaba personală a nivelului. Nivelul poate folosi orice protocol dorește, cu condiția ca acesta să funcționeze (adică să îndeplinească serviciul oferit). Nivelul poate de asemenea să schimbe protocoalele după cum vrea, fără ca acest lucru să afecteze programele din nivelurile superioare.

Aceste idei se potrivesc foarte bine cu ideile moderne referitoare la programarea orientată pe obiect. Un obiect, ca și un nivel, posedă un set de metode (operații) care pot fi invocate de către procese din afara obiectului. Semanticele acestor metode definesc mulțimea de servicii pe care le oferă obiectul. Parametrii și rezultatele metodelor formează interfața obiectului. Codul intern al obiectului reprezintă protocolul său și nu este vizibil și nici important în afara obiectului.

Deși lumea a încercat ulterior să îl readapteze pentru a fi mai asemănător modelului OSI, modelul TCP/IP nu a făcut inițial o distincție clară între serviciu, interfață și protocol. De exemplu, singurele servicii veritabile oferite de nivelul internet sunt SEND IP PACKET și RECEIVE IP PACKET.

În consecință, protocoalele din modelul OSI sunt mai bine ascunse decât în modelul TCP/IP și pot fi înlocuite relativ ușor pe măsură ce se schimbă tehnologia. Capacitatea de a face asemenea modificări reprezintă unul din scopurile principale ale organizării protocoalelor pe niveluri în modelul OSI.

Modelul de referință OSI a fost conceput *înainte* să fie inventate protocoalele corespunzătoare. Ordinea respectivă semnifică faptul că modelul nu a fost orientat către un set specific de protocoale, fiind prin urmare destul de general. Reversul este că proiectanții nu au avut multă experiență în ceea ce privește acest subiect și nu au avut o idee coerentă despre împărțirea funcțiilor pe niveluri.

De exemplu, nivelul legătură de date se ocupa inițial numai cu rețelele punct-la-punct. Atunci când au apărut rețelele cu difuzare, a trebuit să fie introdus în model un subnivel nou. Când lumea a început să construiască rețele reale utilizând modelul OSI și protocoalele existente, s-a descoperit că acestea nu se potriveau cu specificațiile serviciului cerut (minunea minunilor), astfel că a trebuit introdusă în model convergența subnivelurilor, ca să existe un loc pentru a glosa pe marginea diferențelor. În sfârșit, comitetul se aștepta inițial ca fiecare țară să aibă câte o rețea care să fie în custodia guvernului și să folosească protocoalele OSI, așa că nu s-a dat nici o atenție interconectării. Pentru a nu mai lungi povestea, să spunem doar că lucrurile s-au petrecut altfel.

În ceea ce privește TCP/IP, lucrurile stau exact pe dos: mai întâi au apărut protocoalele, iar modelul a fost de fapt doar o descriere a protocoalelor existente. Cu protocoalele respective nu era nici o problemă: ele se potriveau perfect cu modelul. Singurul necaz era că *modelul* nu se potrivea cu nici o altă stivă de protocoale. Prin urmare, modelul nu a fost prea util pentru a descrie alte rețele non-TCP/IP.

Pentru a ne întoarce de la subiectele filosofice la subiecte mai specifice, o diferență evidentă între cele două modele se referă la numărul de niveluri: modelul OSI are șapte niveluri, iar TCP/IP are patru. Ambele modele au niveluri (inter-)rețea, transport și aplicație, dar restul nivelurilor sunt diferite.

O altă deosebire privește subiectul comunicației fără conexiuni față de cel al comunicației orientată pe conexiuni. Modelul OSI suportă ambele tipuri de comunicații la nivelul rețea, dar numai comunicații orientate pe conexiuni în nivelul transport, unde acest fapt are importanță (pentru că serviciul de transport este vizibil utilizatorilor). Modelul TCP/IP are numai un mod (fără conexiuni) la nivelul rețea, dar suportă ambele moduri la nivelul transport, ceea ce lasă utilizatorilor posibilitatea alegerii. Această alegere este importantă în mod special pentru protocoale întrebare-răspuns simple.

1.4.4 O critică a modelului și protocoalelor OSI

Nici modelul și protocoalele OSI și nici modelul și protocoalele TCP/IP nu sunt perfecte. Asupra lor se pot formula, și s-au formulat, câteva critici. În prezenta și în următoarea secțiune vom vedea unele dintre aceste critici. Vom începe cu OSI, după care vom examina TCP/IP.

La momentul când a fost publicată a doua ediție a acestei cărți (1989), majoritatea experților în domeniu credeau că modelul și protocoalele OSI se vor impune peste tot și vor elimina orice concurent. Acest lucru nu s-a întâmplat. De ce? O privire spre lecțiile trecutului poate fi utilă. Aceste lecții pot fi rezumate astfel:

1. Ratarea momentului.
2. Tehnologii proaste.
3. Implementări proaste.
4. Politici proaste.

Ratarea momentului

Să vedem mai întâi prima problemă: ratarea momentului. Momentul la care se stabilește un standard este absolut critic pentru succesul acestuia. David Clark de la M.I.T. are o teorie asupra standardelor pe care o numește *Apocalipsa celor doi elefanți* și care este ilustrată în fig. 1-23.

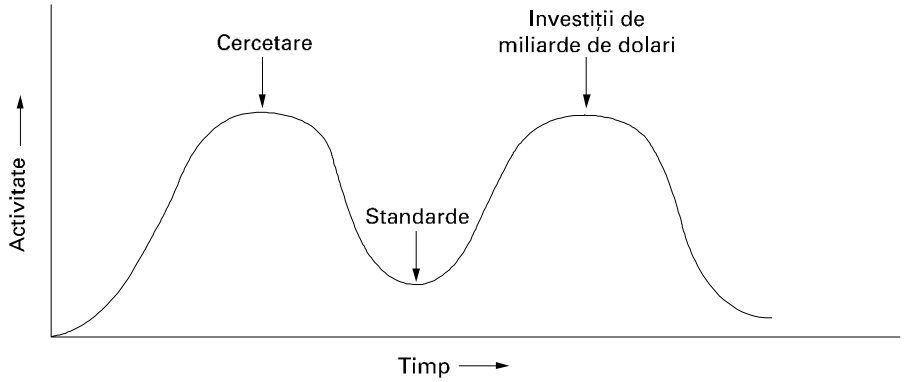


Fig. 1-23. Apocalipsa celor doi elefanți.

Această figură arată volumul de activitate desfășurată în jurul unui subiect nou. Când subiectul este lansat, are loc o explozie a activității de cercetare sub formă de discuții, articole și întâlniri. După un timp, cercetarea se reduce foarte mult, subiectul este descoperit de companii și piața cunoaște un val de investiții de miliarde de dolari.

Este esențial ca standardele să fie definite în intervalul dintre cei doi „elefanți". Dacă ele sunt definite prea devreme, înainte să se încheie cercetarea, atunci subiectul poate să nu fie încă destul de bine înțeles, ceea ce conduce la standarde proaste. Dacă ele sunt definite prea târziu, atunci probabil că atât de multe firme au făcut deja investiții majore realizând lucrurile altfel, încât standardele sunt efectiv ignorate. Dacă intervalul dintre cei doi elefanți este foarte scurt (pentru că toată lumea arde de nerăbdare să treacă la lucru), atunci cei care dezvoltă standardele pot fi prinși la mijloc și striviți.

Acum se vede că protocoalele OSI standard au fost strivite. La momentul apariției lor, protocoalele concurente TCP/IP erau deja folosite pe scară largă în universități, în cercetare. Înainte să vină valul investițiilor de miliarde de dolari, piața din domeniul academic era destul de dezvoltată pentru ca multe firme să înceapă, prudent, să ofere produse TCP/IP. Când a apărut OSI, firmele nu au mai vrut, decât forțate, să sprijine o a doua stivă de protocoale, și, prin urmare, n-au apărut nici un fel de oferte inițiale din partea lor. Fiecare firmă aștepta să înceapă celelalte firme, așa că până la urmă n-a mai început nici o firmă și fenomenul OSI nu s-a mai produs niciodată.

Tehnologii proaste

Al doilea motiv pentru care OSI n-a prins niciodată este că atât modelul cât și protocoalele au defecte. Opțiunea pentru șapte niveluri a fost mai mult politică decât tehnică , și două dintre niveluri (sesiune și prezentare) sunt aproape goale, în timp ce alte două (legătura de date și rețea) sunt prea aglomerate.

Modelul OSI, alături de protocoalele și definițiile de servicii asociate, este extraordinar de complex. Atunci când sunt puse unul peste altul, standardele tipărite au o grosime de câțiva zeci de centimetri. Standardele sunt, de asemenea, dificil de implementat și ineficiente în funcționare. În acest context îmi vine în minte o ghicitoare formulată de Paul Mockapetris și citată în (Rose, 1993):

Î: Ce obții când aplici un standard internațional unui gangster?
R: O persoană care îți face o ofertă pe care n-o poți înțelege.

Pe lângă faptul că este incomprehensibil, o altă problemă cu OSI este că unele funcții, cum sunt adresarea, controlul fluxului și controlul erorilor apar repetat în fiecare nivel. Saltzer ș.a. (1994), de exemplu, au arătat că, pentru a fi eficient, controlul erorilor trebuie făcut la nivelul cel mai înalt și că repetarea sa de atâtea ori în nivelurile de mai jos este adesea inutilă și ineficientă.

Implementări proaste

Dată fiind enorma complexitate a modelului și a protocoalelor, nu este de mirare în faptul că implementările inițiale erau uriașe, greoaie și ineficiente. Oricine le încerca se simțea ca opărit. Nu a trecut mult și lumea a asociat „OSI" cu „calitate slabă." Deși odată cu trecerea timpului produsele au devenit mai bune, imaginea s-a deteriorat.

Din contră, una din primele implementări de TCP/IP făcea parte din Berkeley UNIX și era destul de bună (ca să nu mai spunem că era și gratuită). Lumea a început să o folosească repede, ceea ce a determinat apariția unei comunități largi de utilizatori, ceea ce a dus mai departe la îmbunătățiri, iar aceasta a dus la o comunitate și mai numeroasă. În acest caz spirala nu cobora, ci urca.

Politici proaste

Din cauza implementării inițiale, multă lume, în special din mediul academic, a considerat TCP/IP ca o parte din Unix; iar în anii '80 Unix-ul era pentru oamenii din lumea academică cam la fel de popular ca paternitatea (numită apoi incorect maternitate) sau ca plăcinta cu mere.

OSI, pe de altă parte, a fost gândit ca o creație a ministerelor de telecomunicații europene, apoi a Comunității Europene și, mai târziu, a guvernului Statelor Unite. Această viziune s-a dovedit adevărată numai în parte; dar chiar ideea în sine - un grup de birocrați guvernamentali încercând să bage un standard inferior tehnic pe gâtul bieților cercetători și programatori care stau în tranșee și dezvoltă efectiv rețelele de calculatoare - nu a ajutat prea mult. Unii oameni au văzut această abordare în aceeași lumină în care a fost văzut IBM când a anunțat în anii '60 că PL/I era limbajul viitorului, sau DoD care a corectat IBM-ul anunțând că limbajul respectiv era de fapt Ada.

1.4.5 O critică a modelului de referință TCP/IP

Modelul și protocoalele TCP/IP au și ele problemele lor. Mai întâi, modelul nu face o distincție clară între conceptele de serviciu, interfață și protocol. O practică recomandabilă în ingineria programării este să se facă diferența între specificație și implementare, ceea ce OSI face cu multă atenție, pe când TCP/IP nu face. De aceea, modelul TCP/IP nu este un ghid prea bun de proiectare a rețelelor noi folosind tehnologii noi.

În al doilea rând, modelul TCP/IP nu este deloc general și nu este aproape deloc potrivit pentru descrierea altor stive de protocoale în afara celei TCP/IP. De exemplu, descrierea Bluetooth folosind modelul TCP/IP ar fi aproape imposibilă.

În al treilea rând, nivelul gazdă-rețea nu este deloc un nivel - în sensul normal în care este folosit termenul în contextul protocoalelor organizate pe niveluri - ci este o interfață (între nivelurile rețea și legătură de date). Distincția între o interfață și un nivel este crucială și de aceea trebuie să i se acorde atenția cuvenită.

În al patrulea rând, modelul TCP/IP nu distinge (și nici măcar nu menționează) nivelurile fizic și legătură de date. Acestea sunt complet diferite. Nivelul fizic are de-a face cu caracteristicile

transmisiei prin cablu de cupru, fibre optice sau radio. Rolul nivelului legătură de date este să delimiteze începutul și sfârșitul cadrelor și să le transporte dintr-o parte în alta cu gradul de siguranță dorit. Un model corect ar trebui să includă ambele niveluri ca niveluri separate. Modelul TCP/IP nu face acest lucru.

În sfârșit, deși protocoalele IP și TCP au fost atent gândite și bine implementate, multe din celelalte protocoale au fost construite ad-hoc, fiind în general opera câtorva absolvenți care tot „meștereau" la ele până oboseau. Implementările protocoalelor erau apoi distribuite gratuit; ca urmare, ele erau larg utilizate, fără să li se asigure suportul necesar, fiind de aceea greu de înlocuit. Unele protocoale au ajuns acum să fie mai mult o pacoste. Protocolul de terminal virtual, TELNET, de exemplu, a fost proiectat pentru un terminal teletype mecanic de zece caractere pe secundă. Cu toate acestea, 25 de ani mai târziu, protocolul este încă foarte utilizat.

Pentru a rezuma, în pofida acestor probleme, *modelul* OSI (mai puțin nivelurile sesiune și prezentare) s-a dovedit a fi excepțional de util pentru a discuta rețelele de calculatoare. Din contră, *protocoalele* OSI nu au devenit populare. Pentru TCP/IP este adevărată afirmația inversă: *modelul* este practic inexistent, dar *protocoalele* sunt larg utilizate. Dat fiind faptul că informaticienilor le place să prepare - și apoi să și mănânce - propria lor prăjitură, în această carte vom folosi un model OSI modificat, dar ne vom concentra în primul rând pe TCP/IP și alte protocoale înrudite cu el; de asemenea, vom folosi și protocoale mai noi, precum 802, SONET și Bluetooth. Modelul de lucru folosit în carte este modelul hibrid prezentat în fig. 1-24.

Nivelul aplicație
Nivelul transport
Nivelul rețea
Nivelul legătură de date
Nivelul fizic

Fig. 1-24. Modelul hibrid de referință care va fi utilizat în această carte.

1.5 EXEMPLE DE REȚELE

Subiectul rețelelor de calculatoare acoperă diferite tipuri de rețele, mari și mici, arhicunoscute sau mai puțin cunoscute. Ele au scopuri, dimensiuni și tehnologii diverse. În următoarele secțiuni, vom studia câteva exemple, pentru a avea o idee despre varietatea pe care o poate regăsi oricine în domeniul rețelelor de calculatoare.

Vom porni cu Internet-ul, probabil cea mai cunoscută rețea, și vom studia istoria, evoluția și tehnologiile sale. Apoi vom discuta ATM, care este de multe ori utilizată în nucleul rețelelor (telefonice) mari. Din punct de vedere tehnic, este destul de diferită de Internet, ceea ce evidențiază un contrast interesant. Apoi vom introduce Ethernet, dominantă în cazul rețelelor locale. În final, vom studia IEEE 802.11, standardul pentru rețele fără cablu.

1.5.1 Internet

Internet-ul nu este deloc o rețea, ci o colecție vastă de rețele diverse, care utilizează anumite protocoale comune și oferă anumite servicii comune. Este un sistem neobișnuit prin aceea că nu a

fost planificat de nimeni și nu este controlat de nimeni. Pentru a-l înțelege mai bine, să pornim de la începuturi și să vedem cum s-a dezvoltat și de ce. Pentru o istorie foarte reușită a Internet-ului, este recomandată cartea lui John Naughton (2000). Este una dintre acele cărți rare care nu este numai plăcută la citit, dar are și 20 de pagini de ibid. și op.cit. pentru istoricii serioși. Unele dintre materialele de mai jos sunt bazate pe această carte.

Desigur, au fost scrise nenumărate cărți tehnice despre Internet, și despre protocoalele sale de asemenea. Pentru mai multe informații vedeți, de exemplu, (Maufer, 1999).

ARPANET-ul

Povestea începe la sfârșitul anilor 1950. În momentul în care Războiul Rece era la apogeu, DoD (Department of Defense, rom: Departamentul de Apărare al SUA) a vrut o rețea de comandă și control care să poată supraviețui unui război nuclear. La momentul acela, toate comunicațiile militare foloseau rețelele telefonice publice, care erau considerate vulnerabile. Motivul pentru o astfel de părere poate fi observat în fig. 1-25(a). Aici punctele negre reprezintă oficii de comutare, la ele fiind conectate mii de telefoane. Aceste oficii erau, la rândul lor, conectate la oficii de comutare de nivel mai înalt (oficii de taxare), pentru a forma o ierarhie națională cu un nivel scăzut de redundanță. Vulnerabilitatea sistemului consta în aceea că distrugerea câtorva oficii de taxare putea fragmenta sistemul în mai multe insule izolate.

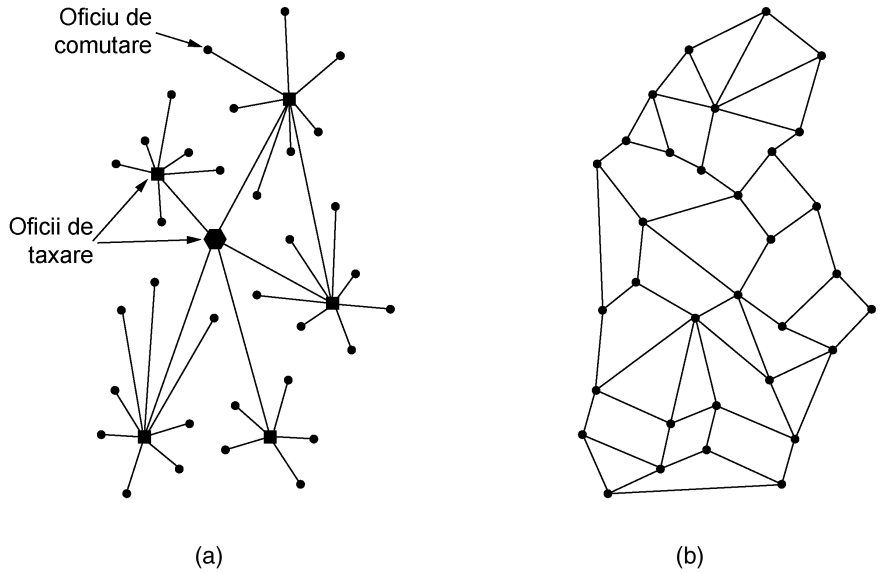


Fig. 1-25. (a) Structura sistemului de telefonie. (b) Sistemul distribuit cu comutare al lui Baran.

În jurul anului 1960, DoD a oferit un contract corporației RAND pentru a găsi o soluție. Unul dintre angajații ei, Paul Baran, a venit cu ideea sistemului distribuit cu un nivel ridicat de toleranța la defecte, prezentat în fig. 1-25(b). Deoarece căile dintre oricare două oficii de comutare erau în acest caz mult mai lungi decât căile pe care semnale analogice puteau să circule fără distorsiuni, Baran a propus utilizarea unei tehnologii digitale cu comutare de pachete prin întregul sistem. Baran a scris câteva rapoarte pentru DoD în care a descris ideile sale în detaliu. Oficialii de la Pentagon au agreat

conceptul și au apelat la AT&T, apoi la monopolul național al telefoniei SUA pentru a construi un prototip. AT&T a desconsiderat imediat ideile lui Baran. Cea mai mare si cea mai bogata companie din lume nu avea de gând să permită unui tânăr oarecare să spună cum să se construiască un sistem de telefonie. Ei au declarat că sistemul propus de Baran nu poate fi construit, și ideea a fost abandonată.

Au mai trecut câțiva ani și DoD încă nu avea un sistem de comandă și control mai bun. Pentru a înțelege ceea ce s-a întâmplat în continuare trebuie să ne întoarcem în Octombrie 1957, când Uniunea Sovietică a întrecut SUA în domeniul spațial prin lansarea primului satelit artificial, Sputnik. Când președintele Eisenhower a încercat să afle cine adormise la comandă, a fost surprins să afle că Armata, Marina și Forțele Aeriene își disputau bugetul de cercetare al Pentagonului. Răspunsul lui imediat a fost crearea unei singure organizații de cercetare în domeniul apărării: ARPA (Advanced Research Projects Agency, rom: Agenția de Cercetare pentru Proiecte Avansate). ARPA nu avea nici oameni de știință, nici laboratoare; de fapt, nu avea decât un birou și un mic buget (după standardele Pentagonului). Își ducea misiunile la îndeplinire prin acordarea de granturi (fonduri pentru cercetare) și contracte universităților și companiilor ale căror idei păreau promițătoare.

În primii câțiva ani, ARPA a încercat să afle care îi era misiunea. În 1967, atenția directorului Larry Roberts a fost atrasă de domeniul rețelelor. A contactat diverși experți ca să decidă ce este de făcut. Unul dintre ei, Weslez Clark, a sugerat construirea unei subrețele cu comutare de pachete, dând fiecărei gazde propriul ruter, așa cum este ilustrat în fig. 1-12.

După un oarecare scepticism inițial, Roberts a adoptat ideea și a prezentat o lucrare destul de vagă despre ea la Simpozionul ACM SIGOPS ținut în Gatlinburg, Tennessee la sfârșitul lui 1967 (Roberts, 1967). Spre surprinderea lui Roberts, o altă lucrare prezentată la aceeași conferință descria un sistem similar, care nu numai că fusese proiectat, dar fusese și implementat sub comanda lui Donald Davies de la NPL (National Physical Laboratories, rom: Laboratoarele Naționale de cercetări în Fizică), Anglia. Sistemul propus de NPL nu era un sistem național (conecta numai câteva calculatoare în campusul NPL) dar demonstrase că comutarea de pachete poate fi funcțională. În plus, cita din rapoartele timpurii ale lui Baran care fuseseră desconsiderate la momentul respectiv. Roberts s-a întors de la Gatlinburg hotărât să construiască ceva ce urma să devină cunoscut sub numele de ARPANET.

Subrețeaua trebuia să fie formată din minicalculatoare numite **IMP**-uri (**Interface Message Processors** - procesoare de mesaje de interfață) conectate prin linii de transmisie. Pentru o siguranță mare, fiecare IMP trebuia legat la cel puțin alte două IMP-uri. Subrețeaua avea să fie o subrețea datagramă, astfel că dacă unele linii și IMP-uri se defectau, mesajele puteau fi redirijate automat pe căi alternative.

Fiecare nod al rețelei era format dintr-un IMP și dintr-o gazdă, aflate în aceeași încăpere și legate printr-un fir scurt. O gazdă putea să trimită mesaje de până la 8063 biți spre IMP-ul său, iar acesta descompunea apoi mesajele în pachete de cel mult 1008 biți și le retransmitea la destinație separat. Fiecare pachet era primit în întregime înainte de a fi reexpediat, astfel că subrețeaua a fost prima rețea electronică memorează-și-retransmite cu comutare de pachete.

ARPA a căutat apoi o ofertă pentru construirea subrețelei. Au depus oferte douăsprezece firme. După evaluarea tuturor propunerilor, ARPA a selectat BBN, o firmă de consultanță din Cambridge, Massachusetts, și în 1968 a încheiat cu aceasta un contract pentru construirea subrețelei și scrierea programelor de subrețea. BBN a decis să utilizeze pe post de IMP-uri minicalculatoare Honeywell DDP-316 special modificate, dispunând de o memorie internă de 12K cu cuvinte pe 16 biți. IMP-urile nu aveau discuri, pentru că părțile mobile erau considerate nesigure. IMP-urile au fost interco-

nectate prin linii de 56 Kbps închiriate de la companii de telefoane. Deși 56 Kbps este acum o variantă pentru adolescenții care nu își permit ADSL sau cablu, era cea mai bună alternativă a momentului respectiv.

Programele au fost împărțite în două: pentru subrețea și pentru gazde. Programele de subrețea cuprind gestionarea capătului dinspre IMP al conexiunii gazdă-IMP, protocolul IMP-IMP și un protocol sursă IMP - destinație IMP, proiectat pentru a mări siguranța. Proiectul inițial al rețelei ARPANET este prezentat în fig. 1-26.

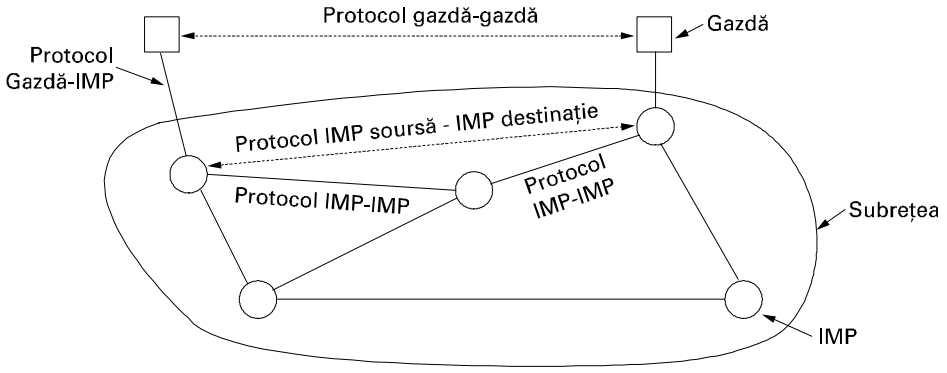


Fig. 1-26. Proiectul inițial al rețelei ARPANET.

Și în afara subrețelei erau necesare programe: gestionarea capătului dinspre gazdă al conexiunii gazdă-IMP, protocolul gazdă-gazdă și programe de aplicație. În scurt timp, a devenit clar că BBN considera sarcina sa încheiată din momentul în care acceptase un mesaj pe un fir gazdă-IMP și îl plasase pe firul gazdă-IMP destinație.

Robets avea o nouă problemă: gazdele aveau și ele nevoie de programe. Pentru a rezolva aceasta problemă, el a convocat o adunare a cercetătorilor în rețele, majoritatea fiind tineri absolvenți de facultate, la Snowbird, în Utah, în vara anului 1969. Absolvenții se așteptau ca niște experți în rețele să le explice proiectarea și software-ul rețelei și ca fiecare din ei să primească după aceea sarcina de a scrie o parte din programe. Au rămas însă muți de uimire când au constatat că nu exista nici un expert în rețele și nici o proiectare serioasă. Trebuiau să își dea seama singuri ce au de făcut.

Cu toate acestea, în decembrie 1969 începea deja să funcționeze o rețea experimentală cu patru noduri, la UCLA, UCSB, SRI și Universitatea din Utah. Au fost alese aceste patru instituții pentru că toate aveau un număr mare de contracte cu ARPA și toate aveau calculatoare gazdă diferite și complet incompatibile (doar ca treaba să fie mai amuzantă). Pe măsură ce se aduceau și se instalau mai multe IMP-uri, rețeaua creștea rapid; în scurt timp, s-a întins pe tot spațiul Statelor Unite. Fig. 1-27 arată cât de repede a crescut ARPA în primii 3 ani.

Pe lângă ajutorul oferit pentru dezvoltarea tânărului ARPANET, ARPA a finanțat de asemenea cercetări în domeniul rețelelor de sateliți și rețelelor mobile radio cu pachete. Într-o faimoasă demonstrație, un camion care circula în California folosea rețeaua radio cu pachete pentru a trimite mesaje către SRI, aceste mesaje erau retransmise apoi prin ARPANET pe Coasta de Est, iar de aici mesajele erau expediate către University College din Londra prin rețeaua de sateliți. Acest lucru permitea unui cercetător din camion să utilizeze un calculator din Londra în timp ce călătorea prin California.

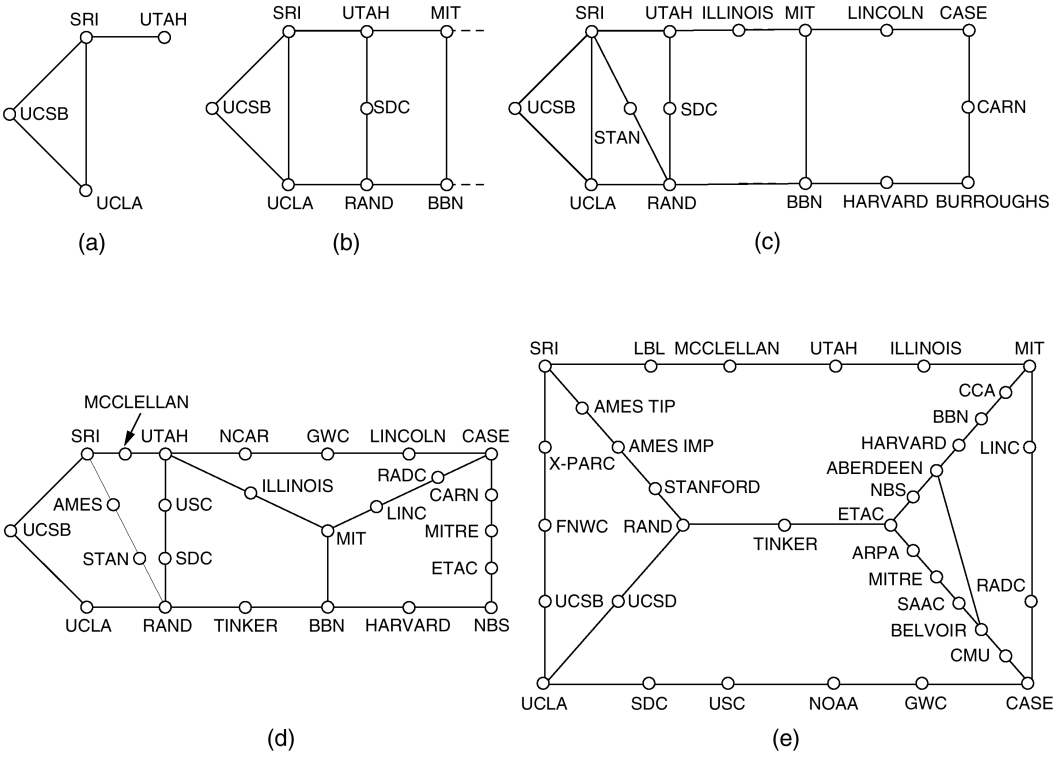


Fig. 1-27. (a) Dec.. 1969. (b) Iulie 1970. (c) Martie 1971. (d) Aprilie 1972. (e) Sept. 1972.

Acest experiment a demonstrat totodată că protocoalele ARPANET existente nu erau potrivite pentru a rula pe mai multe rețele. Observația a condus la noi cercetări asupra protocoalelor, culminând cu invenția modelului și protocoalelor TCP/IP (Cerf și Kahn, 1974). TCP/IP a fost proiectat special pentru a trata comunicarea prin inter-rețele, un lucru care devenea din ce în ce mai important, pe măsură ce tot mai multe rețele erau legate la ARPANET.

Pentru a încuraja adoptarea acestor noi protocoale, ARPA a semnat câteva contracte cu BBN și cu University of California din Berkeley pentru a integra protocoalele în Berkeley UNIX. Cercetătorii de la Berkeley au dezvoltat o interfață de programare a rețelei (soclurile) și au scris numeroase aplicații, utilitare și programe de administrare care să simplifice interconectarea.

Momentul era ideal. Multe universități tocmai achiziționaseră un al doilea sau al treilea calculator VAX și un LAN care să le conecteze, dar nu aveau nici un fel de programe de interconectare. Când a apărut 4.2BSD, cu TCP/IP, socluri și multe utilitare de rețea, pachetul complet a fost adoptat imediat. Mai mult chiar, folosind TCP/IP, LAN-urile se puteau lega simplu la ARPANET și multe LAN-uri au făcut acest lucru.

În anii ‘80 au fost conectate la ARPANET multe alte rețele, în special LAN-uri. Pe măsură ce creștea dimensiunea rețelei, găsirea gazdelor devenea tot mai costisitoare; de aceea, a fost creat **DNS** (**Domain Name System**, rom: Sistemul Numelor de Domenii), care organiza mașinile în domenii și punea în corespondență numele gazdelor cu adrese IP. De atunci încoace, DNS a ajuns să

fie un sistem de baze de date distribuit, generalizat, folosit pentru a memora diverse informații referitoare la procedurile de atribuire a numelor. Vom studia detaliat acest sistem în cap. 7.

NSFNET

La sfârșitul anilor 1970, NSF (U.S. National Science Foundation, rom: Fundația Națională de Știință din SUA) a remarcat impactul imens pe care ARPANET-ul îl avea asupra cercetării universitare, rețeaua permițând savanților din toată țara să partajeze date și să colaboreze la proiecte de cercetare. Dar, pentru a se conecta la ARPANET, o universitate trebuia să aibă un contract de cercetare cu DoD, iar multe universități nu aveau. Răspunsul NSF a fost proiectarea unui succesor al ARPANET care să fie deschis tuturor grupurilor de cercetare din universități. Pentru a avea ceva concret de la care să pornească, NSF a decis să construiască o rețea tip coloană vertebrală (backbone) pentru a conecta cele 6 centre de supercalculatoare pe care le deținea în San Diego, Boulder, Champaign, Pittsburgh, Ithaca, Princeton. Fiecărui calculator i s-a dat un frate mai mic, care era de fapt un microcalculator LSI-11 denumit **fuzzball**. Aceste fuzzball-uri erau conectate cu linii închiriate de 56 Kbps și formau o subrețea , care folosea aceeași tehnologie ca și ARPANET. Tehnologia programelor era însă diferită: fuzzball-urile au fost proiectate pentru a conversa direct folosind TCP/IP, ceea ce a condus la crearea primei rețele pe arie larga bazată pe TCP/IP (TCP/IP WAN).

NSF a finanțat, de asemenea, un număr de (aproximativ 20, până la urmă) rețele regionale care se conectau la coloana vertebrală, permițând utilizatorilor din mii de universități, laboratoare de cercetare, biblioteci și muzee să acceseze oricare dintre supercalculatoare și să comunice între ei. Rețeaua completă, care includea coloana vertebrală și rețelele regionale, a fost numită **NSFNET**. Aceasta a fost conectată la ARPANET printr-o legătură între un IMP și un fuzzball din laboratorul de la Carnegie-Mellon. Prima coloană vertebrală NSFNET este ilustrată în fig. 1-28.

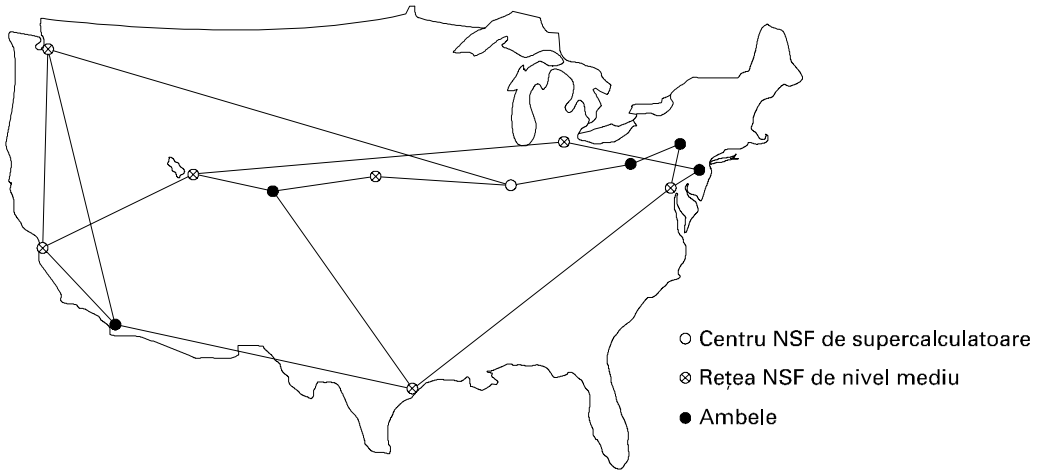


Fig. 1-28. Coloana vertebrală NSFNET în 1988.

NSFNET-ul a reprezentat un succes foarte rapid și a fost suprasolicitat din clipa în care a început să funcționeze. NSF a început imediat să planifice succesorul NSFNET-ului și a semnat un contract cu consorțiul MERIT cu sediul în Michigan. Pentru realizarea coloanei vertebrale numărul 2, au fost închiriate de la MCI (care a fuzionat între timp cu WorldCom) canale cu fibre optice de 448 Kbps. Ca rutere s-au folosit IBM PC-RT. Și această rețea a devenit curând supraîncărcată, drept care, în 1990, a doua coloană vertebrală a fost adusă la viteza de 1.5 Mbps.

Dar creșterea a continuat, iar NSF a realizat că guvernul nu poate finanța interconectările la nesfârșit. În plus, o serie de organizații comerciale erau interesate să se conecteze, dar statutul NSF le interzicea să se lege la rețele finanțate de NSF. În consecință, NSF a încurajat MERIT, MCI și IBM să formeze o corporație nonprofit, **ANS** (**Advanced Networks and Services**, rom: rețele și servicii avansate), ca un pas pe drumul spre comercializare. În 1990, ANS a preluat NSFNET și a înlocuit legăturile de 1.5 Mbps cu legături de 45 Mbps, formând **ANSNET**. Această rețea a funcționat timp de 5 ani și apoi a fost cumpărată de America Online. Dar până atunci, diverse companii ofereau deja servicii IP comerciale și era clar că guvernul trebuia să se retragă din afacerea cu rețele.

Ca să ușureze tranziția și ca să fie sigur că orice rețea regională putea comunica cu orice altă rețea regională, NSF a semnat contracte cu patru operatori de rețele diferiți în vederea stabilirii unui **NAP** (**Network Access Point**, rom: punct de acces la rețea). Acești operatori erau PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.), și Sprint (New York City, unde - din rațiuni legate de NAP - Pennsauken, N.J. se consideră New York City). Fiecare operator de rețea care dorea să ofere servicii de infrastructură pentru rețelele regionale NSF trebuia să se lege la toate NAP-urile.

De aceea, pentru a ajunge de la NAP-ul său la NAP-ul destinației, un pachet trimis din orice rețea regională putea opta între mai multe companii care oferă servicii de transmisie pe coloana vertebrală. În consecință, pentru a fi alese de rețelele regionale, companiile de comunicații au fost forțate să intre în competiție pe baza serviciilor și prețurilor practicate - bineînțeles, aceasta era ideea. Ca rezultat, conceptul unei singure rețele de tip coloană vertebrală a fost înlocuit de o infrastructură competitivă condusă de criterii comerciale. Multora le place să critice Guvernul Federal pentru că nu este destul de inovator, dar în zona rețelelor, DoD și NSF au fost cele care au creat infrastructura care a stat la bazele formării Internet-ului și apoi a cedat-o industriei pentru operare și exploatare.

În timpul anilor 1990, multe alte țări și regiuni construiesc și ele rețele naționale, de multe ori modelate chiar după ARPANET și NSFNET. Acestea includ EuropaNET și EBONE în Europa, care au pornit cu linii de 2 Mbps și apoi au avansat până la linii de 34 Mbps. În cele din urmă, și infrastructura de rețea din Europa a fost cedată industriei spre operare și exploatare.

Folosirea Internet-ului

Numărul rețelelor , mașinilor și utilizatorilor conectați la ARPANET a crescut rapid după ce TCP/IP a devenit, la 1 ian. 1983, unicul protocol oficial. Când au fost conectate NSFNET și ARPANET, creșterea a devenit exponențială. S-au alăturat multe rețele regionale și s-au realizat legături cu rețele din Canada, Europa și Pacific.

Cândva, pe la mijlocul anilor 1980, lumea a început să vadă colecția de rețele ca fiind un internet, iar apoi ca fiind Internet-ul; nu a existat însă nici un toast oficial cu politicieni desfăcând sticle de șampanie.

Substanța care ține legat Internet-ul este modelul de referință TCP/IP și stiva de protocoale TCP/IP. TCP/IP face posibile serviciile universale, putând fi comparată cu adoptarea lățimii standard pentru căile ferate în secolul 19 sau cu adoptarea protocoalelor comune de semnalizare de către toate companiile telefonice.

Ce înseamnă de fapt să fii pe Internet? Definiția noastră este că o mașină este pe Internet dacă folosește stiva de protocoale TCP/IP, are o adresă IP și are posibilitatea de a trimite pachete IP către toate celelalte mașini de pe Internet. Simpla posibilitate de a trimite și primi poștă electronică nu este suficientă, deoarece poșta electronică este redirectată către multe rețele din afara Internet-ului. Oricum, subiectul este cumva umbrit de faptul că milioane de calculatoare personale pot să apeleze

un furnizor de servicii Internet folosind un modem, să primească o adresă IP temporară și apoi să trimită pachete IP spre alte gazde. Are sens să privim asemenea mașini ca fiind pe Internet numai atâta timp cât ele sunt conectate la ruterul furnizorului de servicii.

Tradițional (însemnând din 1970 până în jurul lui 1990), Internet-ul și predecesorii săi au avut patru aplicații principale, după cum urmează:

1. **Poșta electronică**. Facilitatea de a compune, trimite și primi poștă electronică a existat din primele zile ale ARPANET-ului și este extrem de populară. Mulți oameni primesc zeci de mesaje pe zi și consideră poșta electronică principalul lor mijloc de a interacționa cu lumea exterioară, depășind de departe telefonul și poșta obișnuită. Programele de poștă electronică sunt astăzi disponibile practic pe orice tip de calculator.
2. **Știri**. Grupurile de știri sunt forumuri specializate în care utilizatorii cu un anumit interes comun pot să facă schimb de mesaje. Există mii de grupuri de știri, pe subiecte tehnice sau non-tehnice incluzând calculatoarele, știința, divertismentul și politica. Fiecare grup de știri are eticheta, stilul și obiceiurile sale proprii și nenorocirile se vor abate asupra celor care le încalcă.
3. **Conectare la distanță**. Folosind programe ca telnet, rlogin sau ssh, utilizatorii aflați oriunde pe Internet pot să se conecteze la orice mașină pe care au un cont.
4. **Transfer de fișiere**. Copierea fișierelor de pe o mașină din Internet pe alta este posibilă utilizând programul FTP. În acest fel sunt disponibile extrem de multe articole, baze de date și alte informații.

Până la începutul anilor 1990 Internet-ul a fost foarte populat cu cercetători din domeniul academic, guvernamental și industrial. O aplicație nouă, **WWW** (**World Wide Web**), a schimbat total situația și a adus în rețea milioane de noi utilizatori care nu fac parte din mediul academic. Această aplicație, inventată de fizicianul Tim Berners Lee de la CERN, nu a modificat nici una din facilitățile existente, în schimb le-a făcut mai ușor de folosit. Împreună cu programul de navigare Mosaic, scris la Centrul Național pentru Aplicațiile Supercalculatoarelor, WWW-ul a făcut posibil ca un sit să pună la dispoziție un număr de pagini de informații conținând text, poze, sunet și chiar video, în fiecare pagină existând legături către alte pagini. Printr-un clic pe o legătură, utilizatorul este imediat transportat la pagina indicată de legătură. De exemplu, multe firme au o pagină principală cu intrări care trimit la alte pagini pentru informații asupra produselor, liste de prețuri, reduceri, suport tehnic, comunicare cu angajații, informații despre acționari și multe altele.

Într-un timp foarte scurt au apărut numeroase alte tipuri de pagini: hărți, tabele cu cotații la bursă, cataloage de bibliotecă, programe radio înregistrate și chiar o pagină care oferă legături spre textele complete ale multor cărți cărora le-au expirat drepturile de autor (Mark Twain, Charles Dickens, etc.). De asemenea, mulți oameni au pagini personale (home pages).

Mare parte din creșterea Internetului în timpul anilor 1990 a fost alimentată de companii denumite **ISP** (Internet Service Providers, rom: Furnizori de Servicii Internet). Acestea sunt companii care oferă utilizatorilor individuali posibilitatea de a apela, de acasă, una dintre mașinile furnizorului și de a se conecta la Internet, obținând în consecință acces la poșta electronica, WWW și alte servicii similare. La sfârșitul anilor 1990, aceste companii au înregistrat zeci de milioane de noi utilizatori în fiecare an, modificând astfel complet caracterul rețelei, care s-a transformat dintr-o rețea academică și militară într-o utilitate publică, precum sistemul de telefonie. Numărul actual al utilizatorilor Internet nu este cunoscut, dar este cu siguranță de ordinul sutelor de milioane la nivel mondial și probabil că va ajunge la un miliard în curând.

Arhitectura Internet

În această secțiune vom încerca să aruncăm o scurtă privire de ansamblu asupra Internet-ului de astăzi. Din cauza multor fuziuni între companiile de telefoane și companiile ISP, apele au devenit tulburi. și este de cele mai multe ori dificil de precizat care sunt atribuțiile fiecăruia, cine ce anume are de făcut. În consecință această descriere va fi simplificată în raport cu realitatea efectivă. Imaginea de ansamblu este prezentată în fig. 1-29. În continuare, vom analiza această figură bucată cu bucată.

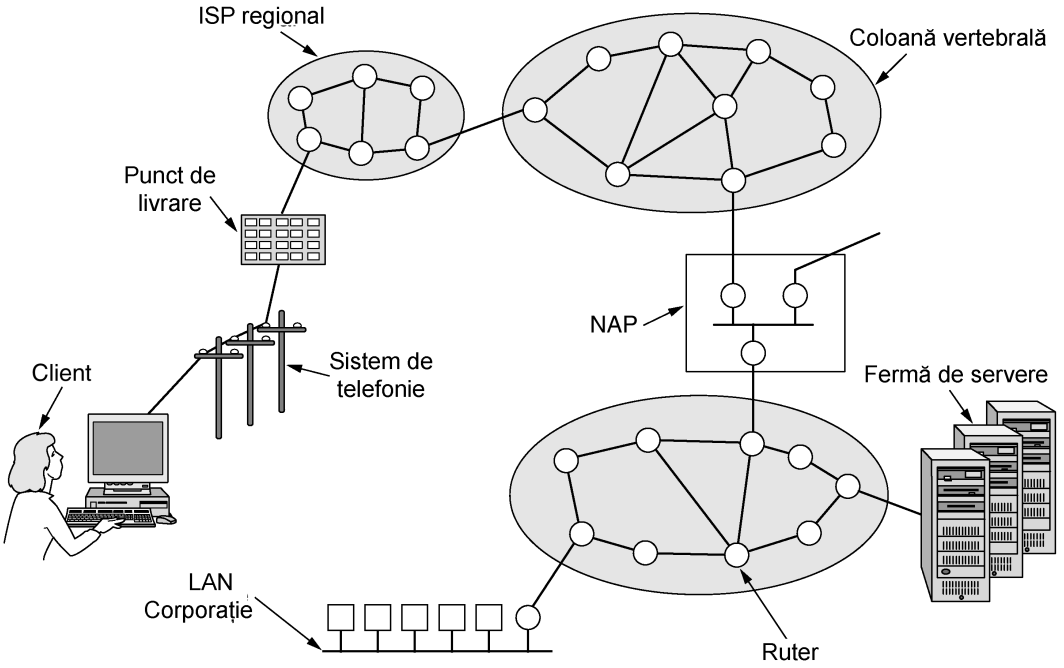


Fig. 1-29. Privire de ansamblu asupra Internet-ului.

Un bun punct de pornire este sistemul propriu al clientului. Să presupunem că acest client sună la ISP-ul său printr-o linie telefonică, așa cum se vede în fig. 1-29. Modemul este o placă din PC-ul clientului care convertește semnalele digitale pe care le produce calculatorul în semnale analogice care pot circula fără probleme prin sistemul telefonic. Aceste semnale sunt transferate la punctul de livrare (POP) al ISP-ului, unde sunt preluate din sistemul telefonic și injectate în rețeaua regionala a ISP. De aici înainte, sistemul este în întregime digital și folosește comutarea de pachete. Daca ISP-ul este același cu furnizorul local de telefonie, punctul de livrare va fi localizat, probabil, chiar în oficiul de comutare al serviciului telefonic, punctul în care se termină firul de telefon al utilizatorului. Chiar dacă ISP-ul nu este același cu furnizorul local de telefonie, punctul de livrare poate fi doar la distanță de câteva oficii de comutare.

Rețeaua regională a ISP este formată prin interconectarea ruterelor din diverse orașe pe care le deservește compania. Dacă pachetul este destinat unei gazde deservite direct de către rețeaua ISP, pachetul va fi livrat direct gazdei. Altfel, el este livrat în continuare operatorului care furnizează companiei ISP servicii de comunicare prin coloana vertebrală (backbone) a rețelei.

În partea superioară a acestei ierarhii sunt operatorii principali de la nivelul de coloană vertebrală a rețelei, companii precum AT&T sau Sprint. Aceștia operează coloane vertebrale mari, internaționale, cu mii de rutere conectate prin fibra optică cu bandă largă de transfer. Corporațiile mari și firmele care oferă servicii de găzduire (hosting), utilizează ferme de servere (mașini care pot servi mii de pagini Web pe secundă) sunt conectate adeseori direct la nivelul coloanei vertebrale. Operatorii încurajează această conectare directă prin închirierea de spații în ceea ce se numește **„hotelul companiei de transport" (carrier hotel)**, și reprezintă de cele mai multe ori **sertare (racks)** pentru echipamente aflate în aceeași cameră cu ruterul, pentru a permite conexiuni scurte și rapide între fermele de servere si coloana vertebrală a rețelei.

Dacă un pachet trimis în coloana vertebrală este destinat unui ISP sau unei companii deservite de aceeași coloană, el este transmis celui mai apropiat ruter. Oricum există multe astfel de coloane vertebrale în întreaga lume, astfel încât un pachet poate să treacă într-o coloană concurentă. Pentru a permite pachetelor să treacă dintr-o coloană în alta, toate aceste coloane principale sunt conectate în NAP-urile (Network Access Point, rom: Punct de acces în rețea) discutate mai devreme. În principiu, un NAP este o cameră plină cu rutere, cel puțin unul pentru fiecare coloană vertebrală conectată. O rețea locală camerei conectează toate aceste rutere, astfel încât pachetele să poată fi retransmise din orice coloană în orice altă coloană. În afară de interconectarea în NAP-uri, coloanele vertebrale de dimensiuni mari au numeroase conexiuni directe între ruterele lor, tehnică denumită **conectare privată (private peering)**. Unul dintre multiplele paradoxuri ale Internet-ului este acela că ISP-urile care sunt la nivel public în competiție pentru clienți, cooperează de cele mai multe ori pentru a realiza astfel de conectări private (private peering) (Metz, 2001).

Astfel se încheie acest scurt tur de orizont asupra Internet-ului. Vom avea multe de spus despre componentele individuale și proiectarea lor, despre algoritmi și despre protocoale în capitolele următoare. Merită de asemenea menționat în trecere că anumite companii și-au interconectat toate rețelele interne existente, folosind de multe ori aceleași tehnologii ca și Internet-ul. Aceste **intranet-uri** sunt accesibile de cele mai multe ori numai din interiorul companiei, dar altfel funcționează la fel ca Internet-ul.

1.5.5 Rețele orientate pe conexiune

Încă de la începuturile domeniului rețelelor, există un război între cei care susțin subrețelele fără conectare (de exemplu datagramele) și cei care susțin subrețelele orientate pe conexiune. Susținătorii subrețelelor fără conexiune provin din comunitatea ARPANET/Internet. Amintiți-vă că dorința inițială a DoD în finanțarea și construirea ARPANET a fost să aibă o rețea care să continue să funcționeze chiar și după ce mai multe lovituri nucleare îndreptate direct împotriva ei au distrus numeroase rutere si linii de transmisie. De aceea, toleranța la defecte se afla pe primele poziții ale listei de priorități; taxarea clienților nu exista pe acea listă. Această abordare a condus la o proiectare fără conexiune în care fiecare pachet era rutat independent de orice alt pachet. Ca o consecință, dacă anumite rutere se defectează în timpul unei sesiuni, nu apare nici o problemă atâta timp cât sistemul se poate reconfigura singur, dinamic, astfel încât pachetele următoare să găsească o rută către destinație, chiar dacă ea este diferită de cea utilizată până la momentul respectiv.

Tabăra celor care susțin rețelele orientate conexiune provine din lumea comunicațiilor pe linii telefonice. În sistemul telefonic, un utilizator trebuie să formeze numărul pe care dorește să îl apeleze și să aștepte formarea unei conexiuni înainte de a vorbi sau de a transmite date. Aceasta fază de conectare stabilește o rută prin sistemul telefonic, rută care va fi menținută până când apelul este în-

cheiat. Toate cuvintele sau pachetele de date urmează aceeași rută. Dacă o linie sau un comutator de pe respectiva cale se defectează, apelul este încheiat forțat. Aceasta proprietate era exact cea care nu convenea deloc Departamentului de Apărare.

De ce sunt companiile organizate astfel? Din două motive:

1. Calitatea serviciilor
2. Facturarea

Prin setarea unei conexiuni în avans, subrețeaua poate rezerva resurse precum zone tampon de memorie sau capacitatea de procesare a procesorului din ruter. Dacă se face o încercare de a iniția un apel și nu se găsesc suficiente resurse disponibile, apelul este rejectat și apelantul primește un fel de semnal de „ocupat". În acest fel, de îndată ce conexiunea a fost stabilită, conexiunea va obține servicii bune din punct de vedere calitativ. Într-o rețea fără conexiune, dacă prea multe pachete ajung la același ruter în același moment, ruterul va fi sufocat și, probabil, va pierde din pachete. Eventual, utilizatorul va observa și le va retrimite, dar calitatea serviciilor va fi proastă și deloc potrivită pentru comunicații audio sau video, cu excepția cazurilor în care rețeaua este doar foarte puțin încărcată. Nu mai este nevoie să precizăm că pentru companii calitatea de transmitere a semnalului audio este un parametru extrem de important, și de aceea preferă rețelele orientate pe conexiune.

Cel de-al doilea motiv pentru care companiile de telefonie preferă serviciile orientate pe conexiune este acela că sunt obișnuite să taxeze utilizatorul în funcție de timpul de conexiune. Atunci când se face un apel la distanță (chiar și local, dar în afara Americii de Nord) taxarea se face la minut. La apariția rețelelor, aceste companii au fost automat atrase în acest sistem, în care taxarea la minut era ușor de făcut. Dacă trebuie stabilită o conexiune înainte de transmisia propriu-zisă a datelor, ceasul de taxare este pornit. Dacă nu există conexiune, nu poți fi taxat pentru ea.

Culmea, menținerea sistemului de taxare este foarte scumpă. Dacă o companie de telefonie ar trebui să adopte o schemă de plată cu rate lunare fixe, fără a ține cont de numărul de apeluri și fără a ține evidența facturărilor pe convorbire, cu siguranță s-ar economisi sume mari de bani, în ciuda creșterii însemnate a numărului de apeluri care va rezulta. Factorii politici, de reglementare și de altă natură sunt însă împotrivă. Destul de interesant este că o astfel de politică este funcțională în alte sectoare. De exemplu, cablul TV este facturat cu o rată lunară fixă, indiferent de cât de mult te uiți la televizor. Ar fi putut să fie proiectat și având la bază un principiu plată-pentru-utilizare (pay-per-view), dar nu s-a făcut așa, în parte și din cauza cheltuielilor impuse de o asemenea strategie de facturare (dată fiind calitatea slabă a majorității televiziunilor, trebuie luat în considerare chiar și factorul „jenă"). Un alt exemplu sunt parcurile tematice care încasează o taxă de intrare zilnică, spre deosebire de caravane, care taxează plimbarea.

Acestea fiind spuse, nu va fi o surpriză că toate rețelele proiectate de industria de telefonie au avut subrețele orientate pe conexiune. Ceea ce este probabil surprinzător este că și Internet-ul deviază în aceasta direcție, pentru a oferi o calitate mai bună pentru serviciile audio și video. Vom reveni la acest subiect în cap. 5. Dar, să examinăm în continuare câteva rețele orientate pe conexiune.

X.25 și Frame Relay (releu de cadre)

Primul exemplu de rețea orientată conexiune este X.25, care a fost prima rețea publică de date. A fost dată în folosință în anii 1970, într-un moment în care serviciile telefonice erau un monopol peste tot, și compania de telefonie din fiecare țară se aștepta să existe și o rețea de date unică în țară – a lor. Pentru a folosi X.25, un calculator a stabilit mai întâi o conexiune cu calculatorul aflat la distanță, adică a făcut un apel telefonic. Pentru această conexiune s-a alocat un număr de conexiune

folosit apoi în transferul pachetelor de date (deoarece pot fi deschise mai multe conexiuni în același timp). Pachetele de date erau foarte simple, fiind formate dintr-un antet de 3 ... 128 de octeți de date. În antet se regăsea un număr de conexiune de 12 biți, un număr de secvență al pachetului, un număr de confirmare pozitivă (ACK) și câțiva biți oarecare. Rețelele X.25 au funcționat aproape un deceniu cu un oarecare succes.

În anii 1980, rețelele X.25 au fost înlocuite pe scară largă cu un nou tip de rețea, denumit **Frame Relay** (Releu de Cadre). În esență, este vorba de o rețea orientată pe conexiune, fără control al erorilor și fără control al fluxului de date. Deoarece era orientată pe conexiune, pachetele erau furnizate în ordine (dacă erau furnizate). Aceste caracteristici – distribuire de pachete în ordine, lipsa de control al erorilor, lipsa de control al fluxului au făcut ca Frame Relay să se asemene cu o rețea locală de dimensiuni mari. Aplicația cea mai importantă a fost interconectarea rețelelor locale aflate în diverse birouri ale companiilor. Deși Frame Relay a avut parte de un succes modest, este folosit și astăzi în anumite companii.

ATM (Asynchronous Transfer Mode)

Încă o rețea orientată pe conexiune – una mult mai importantă de această dată – este ATM (ATM Asynchronous Transfer Mode, rom: Mod de Transfer Asincron). Acest nume, oarecum ciudat, este justificat prin aceea că, în timp ce în rețelele telefonice majoritatea transmisiilor sunt sincrone (strâns legate de un semnal de ceas), în rețelele ATM transmisiile nu sunt sincrone.

ATM a fost proiectat la începutul anilor 1990 și lansat la mijlocul acestei perioade incredibile (Ginsburg, 1996; Goralski, 1995; Ibe, 1997; Kim et al., 1994; at Stallings, 2000). ATM urma să rezolve toate problemele de rețele și telecomunicații ale lumii, unificând transmisiile de voce, date, televiziune prin cablu, telex, telegraf, porumbei mesageri, cutii de conserve conectate prin sfori, semnale cu fum, și orice altceva într-un singur sistem integrat care să poată face totul pentru toată lumea. Nu s-a întâmplat. În mare parte, problemele erau similare cu acelea care au fost descrise mai devreme în ceea ce privește OSI, adică: ratarea momentului, tehnologii slabe, implementări ineficiente, politici proaste. După ce tocmai învinseseră companiile telefonice în runda I, mulți membri din comunitatea Internet au văzut ATM-ul pe poziția Internet-ului în lupta cu companiile mixte telefonie-ISP: Următorul. Dar nu a fost așa, și de această dată chiar și cei mai fanatici susținători ai datagramelor au trebuit să recunoască faptul că Internet-ul lăsa mult de dorit în privința calității serviciilor. Pentru a scurta povestea, ATM a înregistrat un succes mult mai mare decât OSI și este acum utilizat pe scară largă în cadrul sistemelor de telefonie, adeseori vehiculând chiar pachete IP. Deoarece ATM este utilizat la ora actuală de majoritatea companiilor numai pentru operațiile de rutare și transport intern, în cele mai multe cazuri utilizatorii nu sunt conștienți de existența lui, chiar dacă el este operațional.

Circuite virtuale ATM

Deoarece rețelele ATM sunt orientate pe conexiune, transmisia datelor necesită mai întâi transmisia unui pachet pentru inițializarea conexiunii. Pe măsură ce pachetul de inițializare circulă prin subrețea, toate ruterele de pe drumul pe care îl parcurge își creează câte o înregistrare în tabelele de dirijare în care înregistrează existența conexiunii și rezervă resursele necesare pentru ea. Conexiunile sunt de cele mai multe ori denumite circuite virtuale, în analogie cu circuitele fizice utilizate în sistemele de telefonie. Majoritatea rețelelor ATM suportă și circuite virtuale permanente, care sunt conexiuni permanente între două gazde aflate la distanță. Acestea sunt similare cu liniile închiriate din lumea telefoniei. Fiecare conexiune, fie ea temporară sau permanentă, are un identificator de conexiune unic. Un circuit virtual este prezentat în fig. 1-30.

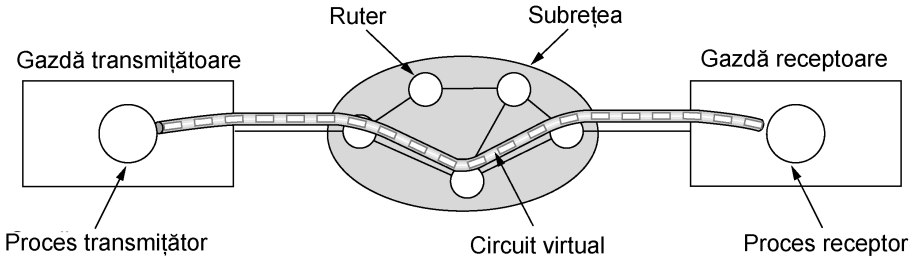


Fig. 1-30. Un circuit virtual

Îndată ce o conexiune a fost stabilită, oricare dintre părți poate să înceapă să transmită date. Ideea de bază în cazul rețelelor ATM este să se transmită toate informațiile în pachete mici, de dimensiune fixă, denumite celule (cells). Celulele au 53 de octeți, din care 5 octeți reprezintă antetul, iar restul de 48 reprezintă încărcătura efectivă, după cum se poate vedea în figura1-31. O parte din antet reprezintă identificatorul de conexiune, astfel încât atât transmițătorul cât și receptorul, precum și toate ruterele intermediare pot ști corespondența dintre celule și conexiuni (care celule aparțin cărei conexiuni). Această informație permite fiecărui ruter să dirijeze fiecare celulă pe care o primește. Dirijarea celulelor este implementată direct în partea hardware a ruterelor și este o operație rapidă. De fapt, argumentul principal în alegerea de celule de dimensiune fixă este acela că este mai ușor de construit partea hardware pentru dirijare dacă ea are de a face cu pachete scurte și egale ca dimensiune. Pachetele IP de lungime variabilă trebuie dirijate de programe (software), proces care este mai lent. Un alt avantaj al rețelelor ATM este acela că partea hardware poate fi configurată să multiplice o celulă pe care o primește la intrare pe mai multe linii de ieșire, o proprietate obligatorie în cazul în care trebuie abordată transmisia unui program de televiziune difuzat către mai mulți receptori. La urma urmei, celulele mici nu blochează nici o linie pentru prea mult timp, ceea ce face garantarea calității serviciilor mai ușoară.

Toate celulele urmează aceeași cale către destinație. Livrarea celulelor nu este garantată, dar ordinea lor da. Dacă doua celule 1 și 2 sunt transmise în această ordine (1,2), dacă amândouă ajung, ele vor ajunge în aceeași ordine, niciodată nu va ajunge 2 înaintea lui 1. Dar oricare dintre ele, sau chiar amândouă se pot pierde pe drum. Este de datoria protocoalelor nivelului superior să repare eroarea cauzată de celulele pierdute. De reținut că, deși această garanție nu este perfectă, este mai bună decât cea pe care o oferă Internet-ul. Acolo nu numai că pachetele se pot pierde, dar și ordinea de ajungere la destinație poate fi oricare (nu are legătură cu ordinea de transmisie).

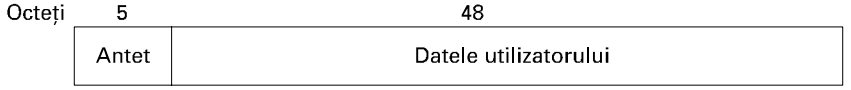


Fig. 1-31. O celulă ATM

Rețelele ATM sunt organizate similar cu rețelele WAN tradiționale, cu linii și comutatoare (rutere). Cele mai des întâlnite viteze de lucru pentru rețelele ATM sunt 155 Mbps și 622 Mbps, deși sunt posibile și viteze mai mari. Viteza de 155 Mbps a fost aleasă pentru că este foarte apropiată de viteza minimă obligatorie pentru transmisia de televiziune cu rezoluție înaltă. Decizia de a alege viteza exactă de 155.52 Mbps a fost făcută pentru compatibilitatea cu sistemul de transmisie

SONET de la AT&T, care va fi studiat în cap. 2. Viteza de 622 Mbps a fost aleasă astfel încât să fie echivalentă cu transmisia simultană a 4 canale de 155 Mbps.

Modelul de referință ATM

ATM are propriul său model de referință, diferit de modelul OSI și diferit de asemenea de modelul TCP/IP. Acest model este ilustrat în fig. 1-32. El constă din trei niveluri - nivelul fizic, nivelul ATM și nivelul de adaptare ATM - plus orice mai vrea utilizatorul să pună deasupra lor.

Nivelul fizic se ocupă de mediul fizic: voltaj, planificare la nivel de biți și diverse alte aspecte. ATM nu prescrie un set particular de reguli, dar spune în schimb că celulele ATM pot fi trimise direct prin cablu sau fibre optice sau pot fi, la fel de bine, împachetate în interiorul datelor din alte sisteme de transmisie. Cu alte cuvinte, ATM-ul a fost proiectat pentru a fi independent de mediul de transmisie.

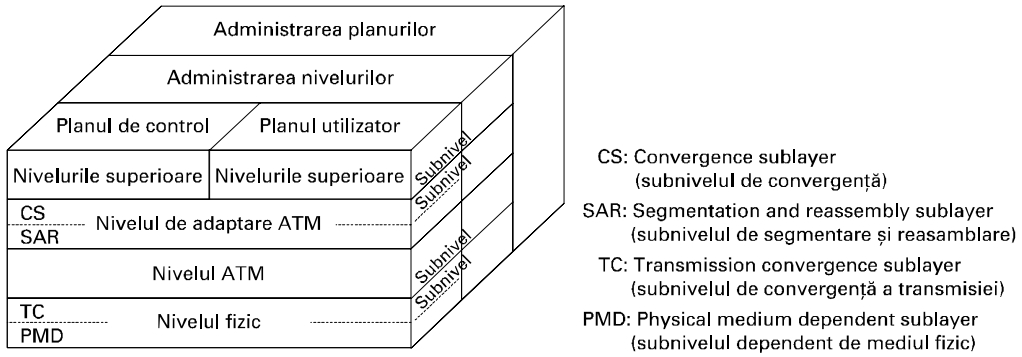


Fig. 1-32. Modelul de referință B-ISDN ATM.

Nivelul ATM se ocupă de celule și de transportul celulelor. Nivelul definește structura unei celule și spune ce reprezintă câmpurile celulelor. Tot el se ocupă și de stabilirea și eliberarea circuitelor virtuale. Controlul congestiei se realizează tot aici.

Deoarece cele mai multe aplicații nu vor să lucreze direct cu celule (deși unele vor), deasupra nivelului ATM a fost definit un nivel care permite utilizatorilor să trimită pachete mai mari decât o celulă. Interfața ATM segmentează aceste pachete, transmite celulele individual și le reasamblează la celălalt capăt. Acest nivel este **AAL** (**ATM Adaption Layer,** rom: nivelul de adaptare ATM).

Spre deosebire de cele două modele de referință anterioare, care erau bidimensionale, modelul ATM este definit ca fiind tridimensional, după cum se arată în fig. 1-32. **Planul utilizator** se ocupă, printre altele, cu transportul datelor, controlul fluxului, corectarea erorilor. Prin contrast, sarcina **planului de control** este să trateze conexiunile. Funcțiile de administrare ale nivelurilor și planurilor se referă la gestionarea resurselor și coordonarea între niveluri.

Fiecare din nivelurile fizic și AAL sunt împărțite în două subniveluri: un subnivel care face munca efectivă, la bază, și un subnivel de convergență, deasupra, care pune la dispoziția nivelului situat peste el interfața adecvată. Funcțiile nivelurilor și subnivelurilor sunt prezentate în fig. 1-33.

Subnivelul **PMD** (**Physical Medium Dependent**, rom: dependent de mediul fizic) asigură interfața cu cablul propriu-zis. Acest subnivel transferă biții și se ocupă de planificarea transmisiei la nivel de biți. În cazul unor companii telefonice și a unor cabluri diferite, subnivelul va fi și el diferit.

Nivel OSI	Nivel ATM	Subnivel ATM	Rol
3/4	AAL	CS	Asigurarea interfeței standard (convergenței)
		SAR	Segmentarea şi reasamblarea
2/3	ATM		Controlul fluxului Generarea/extragerea antetelor din celule Administrarea circuitelor/căilor virtuale Multiplexarea/demultiplexarea celulelor
2	Fizic	TC	Decuplarea ratei celulelor Generarea şi verificarea sumelor de control din antete Generarea celulelor Împachetarea/despachetarea celulelor din plic Generarea cadrelor
1		PMD	Temporizarea biților Accesul fizic la rețea

Fig. 1-33. Nivelurile și subnivelurile ATM și funcțiile acestora.

Celălalt subnivel al nivelului fizic este subnivelul TC (Transmission Convergence, rom: convergența transmisiei). Când sunt transmise celulele, nivelul TC le expediază sub forma unui șir de biți spre nivelul PMD. Acest lucru este ușor de făcut. La celălalt capăt, subnivelul TC primește de la subnivelul PMD un flux de biți. Sarcina sa este să convertească acest flux de biți într-un flux de celule pentru nivelul ATM. Subnivelul TC se ocupă de tot ce este necesar pentru a putea spune unde încep și unde se termină celulele din fluxul de biți. În modelul ATM această funcționalitate este înglobată în nivelul fizic. În modelul OSI și în majoritatea celorlalte rețele, încadrarea, adică transformarea unui flux oarecare de biți într-o secvență de cadre sau de celule, este sarcina nivelului legătură de date. De aceea, în această carte vom discuta funcția respectivă împreună cu nivelul legătură de date, nu cu nivelul fizic.

Așa cum am menționat mai devreme, nivelul ATM gestionează celulele, inclusiv generarea și transportul lor. Mare parte din aspectele interesante ale ATM-ului apar aici. Nivelul ATM este un amestec între nivelurile legătură de date și rețea de la OSI, dar nu este împărțit în subniveluri.

Nivelul AAL este împărțit într-un subnivel **SAR** (**Segmentation And Reassembly**, rom: segmentare și reasamblare) și un subnivel **CS** (**Convergence Sublayer**, rom: subnivel de convergență). Subnivelul inferior descompune pachetele în celule - la capătul la care are loc transmisia - și le recompune la destinație. Subnivelul superior face posibile sistemele ATM care oferă diverse tipuri de servicii pentru diverse aplicații (de exemplu, transferul de fișiere și sistemul video la cerere au cerințe diferite privitoare la gestionarea erorilor, planificare etc.).

Deoarece se preconizează o evoluție descendentă pentru rețelele ATM, ele nu vor fi discutate în continuare în această carte. Oricum, fiind instalate pe scară destul de largă, vor fi în continuare folo-

site pentru câțiva ani buni. Pentru mai multe informații despre ATM, vedeți (Dobrowski și Grise, 2001; Gadeki și Heckart, 1997).

1.5.3 Ethernet

Internet-ul și ATM au fost proiectate pentru WAN. Oricum, multe companii, universități și alte organizații au multe calculatoare care trebuie conectate. Această necesitate a dus la o dezvoltare rapidă a rețelelor locale. În această secțiune vom prezenta câteva lucruri despre cea mai populară dintre rețelele locale, și anume Ethernet.

Povestea începe în primitivul Hawaii la începutul anilor 1970. În acest caz, „primitiv" poate fi interpretat ca „fără sistem de telefonie funcțional". Chiar dacă faptul că nu te deranjează telefonul cât e ziua de lungă poate să facă viața mai plăcută în vacanță, această situație nu era foarte plăcută pentru cercetătorul Norman Abramson și colegii săi de la Universitatea din Hawaii, care încercau să conecteze utilizatorii din mai multe insule aflate la distanță la calculatorul principal din Honolulu. Și cum varianta de a-și trage singuri cablurile pe fundul Oceanului Pacific nu părea viabilă, a trebuit să se caute o altă soluție.

Cea pe care au găsit-o a fost transmisia radio pe unde scurte. Fiecare terminal utilizator era echipat cu un mic sistem radio care avea două frecvențe: Trimite (**upstream** - către calculatorul central) și Primește (**downstream** – de la calculatorul central). Când utilizatorul dorea să contacteze calculatorul, trebuia doar să transmită un pachet care conținea datele pe canalul Trimite. Dacă nu mai transmitea nimeni în acel moment, pachetul ajungea la calculatorul central și i se dădea un răspuns pe canalul Primește. Dacă avea loc o dispută pentru canalul de transmisie, terminalul observa că nu primește confirmarea pozitivă pe canalul de recepție și trimitea din nou. Deoarece era un singur transmițător pe canalul de primire (calculatorul central), aici erau imposibile coliziunile. Acest sistem, care a fost denumit ALOHANET, funcționa destul de bine în condiții de trafic redus, dar eșua de îndată ce traficul pe canalul de Transmisie era aglomerat.

Cam în același timp, un student pe nume Bob Metcalfe și-a obținut diploma de absolvire la M.I.T. și s-a mutat pentru a obține doctoratul la Harvard. În timpul studiilor sale, a ajuns sa cunoască lucrarea lui Abramson. A devenit atât de interesat în acest domeniu încât după ce a absolvit la Harvard, a decis să petreacă vara în Hawaii lucrând împreună cu Abramson, înainte de a începe lucrul la Xerox PARC (Palo Alto Research Center, rom: Centrul de Cercetare de la Palo Alto) . Când a ajuns la PARC, a descoperit că cercetătorii de acolo proiectaseră și construiseră mașinile care mai târziu aveau să fie denumite calculatoare personale. Dar mașinile erau izolate. Folosind cunoștințele pe care le acumulase în timpul lucrului petrecut cu Abramson, a proiectat și implementat – împreună cu colegul său David Boggs – prima rețea locală de calculatoare (Metcalfe și Boggs, 1976).

Au numit sistemul **Ethernet** după *luminiferous ether* (eter), prin care se credea odinioară că se propagă undele electromagnetice (În secolul 19, când fizicianul englez James Clerk Maxwell a descoperit că radiația electromagnetică poate fi descrisă printr-o ecuație de undă, oamenii de știință au presupus că spațiul trebuie să fie umplut cu un mediu eteric prin care aceste radiații se propagau. Numai după faimosul experiment Michelson-Morley din 1887 fizicienii au descoperit că radiația electromagnetică se poate propaga în vid).

Mediul de transmisie în acest caz era un cablu coaxial gros, având o lungime de până la 2.5 km (cu repetoare la fiecare 500m). Până la 256 de mașini pot fi atașate sistemului prin transivere conectate direct în cablu. Un cablu cu mai multe mașini atașate în paralel este numit **cablu multidrop** (multidrop cable). Sistemul funcționa la 2.94 Mbps. O schiță a arhitecturii sale este prezentată în fig.

1-34. Ethernet-ul avea o îmbunătățire majoră față de AOHANET: înainte să transmită, un calculator asculta mediul pentru a vedea dacă nu cumva este altcineva care transmite. Dacă exista deja o transmisie în curs, calculatorul se oprea și aștepta încheierea transmisiei curente. Astfel, se evita interferența cu transmisiunile existente, ceea ce creștea semnificativ eficiența sistemului. ALOHANET nu putea să funcționeze în această manieră pentru că era imposibil pentru un terminal de pe o insulă să detecteze transmisia unui alt terminal de pe o altă insulă. Pe un cablu unic, această problemă era rezolvată.

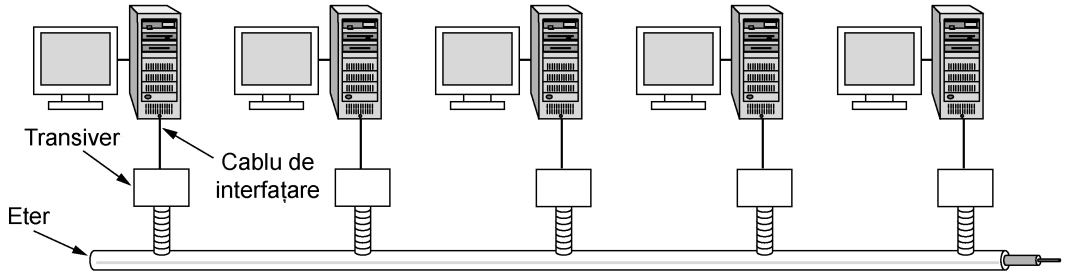


Fig. 1-34. Arhitectura Ethenet-ului original

În ciuda faptului că fiecare calculator asculta mediul înainte să înceapă transmisia, exista în continuare o problemă: ce se întâmplă dacă două calculatoare așteaptă amândouă încheierea transmisiei curente și apoi pornesc propriile transmisii simultan? Soluția este următoarea: fiecare calculator va asculta mediul și în timpul propriei transmisii și dacă detectează interferențe, bruiază linia pentru a anunța toți transmițătorii. Apoi se retrage și așteaptă un interval de timp generat aleator înainte să încerce din nou. Dacă apare o a doua coliziune, timpul de așteptare se dublează, și tot așa, pentru a dispersa (în timp) transmisiile concurente oferind fiecăreia dintre ele șansa de a fi „servită” prima.

Ethernet-ul Xerox a avut un succes atât de mare încât DEC, Intel și Xerox au colaborat pentru a schița un standard pentru o rețea Ethernet de 10 Mbps, denumit standardul DIX. Cu două modificări minore, acesta a devenit standardul IEEE 802.3 în anul 1983.

Din păcate pentru Xerox, compania avea deja reputația de a face invenții (precum calculatorul personal) și apoi să eșueze în valorificarea lor comercială, poveste spusă în *Fumbling the Future* (Smith și Alexander, 1988). Și pentru că Xerox nu a anunțat vreo intenție de a face și altceva cu Ethernet-ul – în afara standardizării lui – Metcalfe și-a format propria companie, 3Com, care urma să producă și să vândă adaptoare Ethernet pentru PC. A vândut peste 100 de milioane.

Ethernet-ul a continuat să se dezvolte și este încă în curs de dezvoltare. Noi versiuni, la 100 Mbps și 1000 Mbps, ba chiar și mai rapide au apărut deja. De asemenea, cablarea s-a îmbunătățit, fiind adăugate și alte facilități, precum comutarea (switching). Vom discuta în detaliu despre Ethernet în cap. 4.

În trecere, merită menționat că Ethernet (IEEE 802.3) nu este singurul standard LAN. Comitetul a standardizat de asemenea Token Bus (Jeton pe Magistrală – 802.4) și Token Ring (Jeton pe Inel – 802.5). Necesitatea de a avea trei standarde mai mult sau mai puțin incompatibile ține mai mult de politică decât de tehnologie. La momentul standardizării, firma General Motors promova o rețea în care topologia era aceeași ca la Ethernet (un cablu liniar), dar calculatoarele obțineau dreptul la transmisie pe rând, prin transmiterea unui scurt pachet denumit **jeton (token)**. Un calculator putea să emită numai dacă era în posesia jetonului, fiind evitate astfel coliziunile. General Motors a

anunțat că această schemă era esențială pentru fabricația de mașini și nu era pregătită să se miște de pe această poziție. Dacă acest anunț nu era susținut, 802.4 nu ar fi existat.

Similar, IBM avea propriul favorit: rețeaua proprietară cu jeton în inel. De această dată, jetonul era transmis prin inel și orice calculator care avea jetonul putea să transmită înainte de a repune jetonul în circulație în inel. Spre deosebire de 802.4, această schemă, standardizată ca 802.5, este încă folosită în birouri și filiale ale IBM, dar practic nicăieri în afara IBM. Oricum, cercetarea avansează către o versiune gigabit, dar pare foarte puțin probabil ca această tehnologie să ajungă la nivelul Ethernet. Pe scurt, chiar dacă a fost cândva un război între Ethernet, Token Ring și Token Bus, Ethernet a câștigat, în special pentru că a fost primul și pentru că oponenții săi nu era destul de buni.

1.5.4 Rețele fără fir: 802.11

Imediat după apariția calculatoarelor portabile, mulți utilizatori visau să intre cu calculatorul portabil personal într-un birou și, miraculos, acesta să fie conectat la Internet. În consecință, mai multe grupuri de studiu am început să caute soluții pentru a atinge acest scop. Cea mai practică abordare era echiparea biroului și a calculatorului cu transmițătoare și emițătoare radio cu rază mică de acțiune pentru a le permite să comunice. Această variantă a dus rapid la comercializarea soluțiilor de rețele locale fără fir de către diverse companii.

Problema era că dintre aceste variante nu se găseau două compatibile. Această proliferare a standardelor însemna că un calculator care era echipat cu un radio marca X nu putea să se conecteze în rețeaua unui birou dacă acesta era echipat cu o stație de la firma Y. În cele din urmă, comunitatea industrială a decis că ar trebui impus un standard pentru LAN fără fir. Astfel, comitetul IEEE care standardizase și LAN-urile cu cablu a primit ca sarcină să schițeze un standard pentru rețele LAN fără fir. Standardul astfel creat s-a numit 802.11. O denumire mai bine cunoscută în argou este **WiFi**. Este un standard important și merită tot respectul, astfel că ne vom referi la el cu numele oficial, 802.11.

Standardul propus trebuia să lucreze în două moduri:

1. În prezența unei stații de bază
2. În absența unei stații de bază

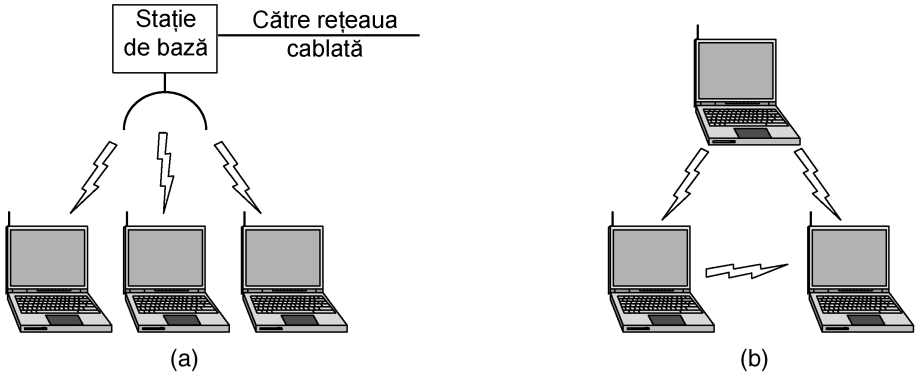


Fig. 1-35. (a) Rețele fără fir cu stație de bază. (b) Conectare ad-hoc.

În primul caz, toate comunicațiile urmau să aibă loc prin intermediul stației de bază, denumită **punct de acces** (**access point**) 802.11. În cel de-al doilea caz, calculatoarele urmau să comunice direct unul cu celălalt. Acest mod este uneori denumit **conectare ad-hoc** (**ad-hoc networking**). Un exemplu tipic este cel al utilizatorilor care se află într-o cameră care nu este echipată cu o stație de bază, calculatoarele lor comunicând direct. Aceste două moduri sunt ilustrate în fig. 1-35.

Prima decizie a fost cea mai simplă: cum să se numească. Toate celelalte standarde LAN aveau numere cum sunt 802.1, 802.2, 802.3, până la 802.10. Așa că noul standard de LAN fără fir s-a numit 802.11. Restul a fost mai dificil de realizat.

În particular, câteva dintre obiectivele care trebuiau atinse erau : găsirea unei benzi de frecvențe care să fie disponibilă, de preferință la nivel mondial; tratarea faptului că semnalele radio au o rază de acțiune limitată; asigurarea menținerii confidențialității utilizatorului; tratarea problemei duratei limitate de lucru a bateriei; considerarea eventualelor efecte pe care sistemul le putea avea asupra oamenilor (provoacă undele radio cancer?); înțelegerea implicațiilor portabilității calculatoarelor; și, în final, construirea unui sistem cu lărgime de bandă suficientă pentru a fi viabil din punct de vedere economic.

La momentul în care s-a început procesul de standardizare (la mijlocul anilor 1990), Ethernet-ul domina deja domeniul rețelelor locale, așa încât comitetul a decis să facă noul standard 802.11 compatibil Ethernet începând de deasupra nivelului legătură de date. Mai exact, ar trebui să se poate transmite un pachet IP într-un LAN fără fir în aceeași manieră în care un pachet IP este transmis prin Ethernet. Desigur, la nivelurile Fizic și Legătură de date apar anumite diferențe inerente față de Ethernet și ele trebuie considerate de către standard.

Mai întâi, un calculator din Ethernet va asculta eterul înainte de a transmite. Numai dacă acesta este liber calculatorul va începe transmisia. În cazul rețelelor LAN fără fir, această idee nu funcționează prea bine. Pentru a vedea de ce, analizați fig. 1-36. Să presupunem că A transmite către B, dar raza de acțiune a lui A este prea mică pentru a îl acoperi și pe C. Atunci când C vrea să transmită, el poate asculta mediul înainte să înceapă, dar faptul că nu aude nimic nu înseamnă că transmisia lui va reuși. Standardul 802.11 trebuia să rezolve și această problemă.

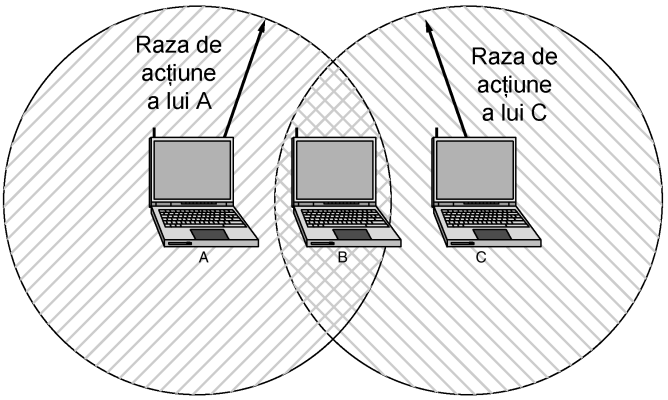


Fig.1-36. Raza de acțiune a unui singur radio poate să nu acopere întregul sistem.

O a doua problemă care trebuia rezolvată era aceea că semnalul radio poate fi reflectat de anumite obiecte solide și deci poate fi recepționat de mai multe ori (pe diverse căi). Această interferență duce la ceea ce se numește **disipare pe mai multe căi** (**multipath fading**).

Cea de-a treia problemă era că o mare parte din aplicații nu erau conștiente de mobilitatea calculatoarelor. De exemplu, multe dintre editoarele de texte aveau o listă de imprimante dintre care una putea fi aleasă pentru tipărirea documentului. Atunci când calculatorul rulează în afara mediului său obișnuit, într-un mediu nou, lista de imprimante implicite nu mai este validă.

Cea de-a patra problemă se referea la mutarea calculatorului portabil din raza de acțiune a unei stații de bază în raza altei stații de bază. Într-un fel sau altul, trebuie găsită o soluție de predare/primire între cele două stații de bază. Deși această problemă apare și la nivelul telefoanelor mobile, ea nu apare la Ethernet și nu avea o soluție la momentul respectiv. Mai exact, rețeaua constă din mai multe celule, fiecare cu propria stație de bază, conectate prin Ethernet, după cum se poate vedea în fig. 1-37. Din exterior, sistemul trebuie să arate ca o singură rețea Ethernet. Conexiunea dintre sistemele 802.11 și lumea exterioară se numește **portal** (**portal**).

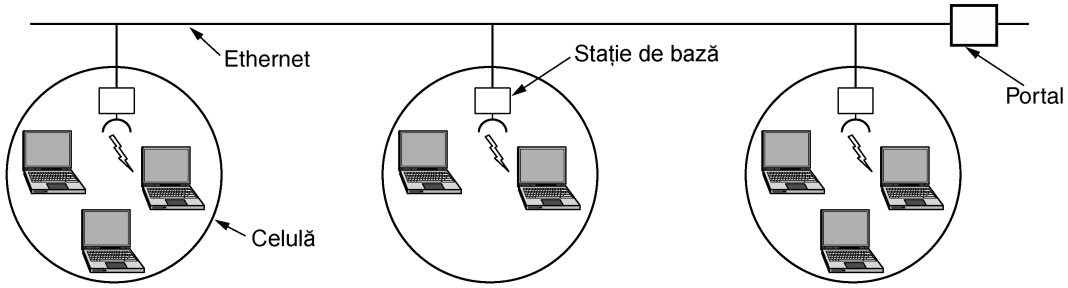


Fig.1-37. O rețea 802.11 cu mai multe celule

După o oarecare muncă, comitetul a obținut o variantă de standard în 1997, variantă care adresa aceste probleme și altele asemănătoare. Rețelele locale fără fir pe care standardul le propunea puteau funcționa la 1 Mbps sau 2 Mbps. Aproape imediat, utilizatorii au început să se plângă de viteza prea scăzută și s-a pornit o nouă campanie pentru obținerea unor standarde mai rapide. În cadrul comitetului a avut loc o ruptură, ceea ce a dus la apariția a două standarde în 1999. Standardul 802.11a folosește o bandă de frecvență mai largă și poate ajunge la viteze mai mari de 54 Mbps. Standardul 802.11b folosește aceeași banda ca și 802.11, dar folosește o tehnică de modulare diferită și poate ajunge la 11 Mbps. Unii văd în aceasta un amănunt important la nivel psihologic, pentru că 11 Mbps este o viteză mai mare decât a Ethernet-ului original, cu cablu. Este foarte probabil ca standardul original 802.11 de 1 Mbps să moară în curând, dar nu se știe care dintre noile standarde va ieși învingător.

Pentru a face lucrurile încă mai complicate decât erau, comitetul 802 a venit cu o nouă variantă, 802.11g, care folosește tehnica de modulare folosită și de 802.11a, dar banda de frecvență a lui 802.11b. Vom reveni în detaliu la 802.11 în cap. 4.

Faptul ca 802.11 urmează să ducă la o revoluție în lumea calculatoarelor și a accesului la Internet este mai presus de orice îndoială. Aeroporturile, gările, hotelurile, magazinele mari și universitățile îl implementează foarte curând. Chiar si cafenelele aflate într-o perioadă de creștere a afacerilor instalează rețele 802.11 pentru ca grupurile de tineri rebeli să poată naviga pe Web în timp ce își savurează cafelele cu lapte. Este foarte probabil ca 802.11 să aibă asupra Internet-ului același efect pe care l-au avut portabilele în lumea calculatoarelor: să-l facă mobil.

1.6 STANDARDIZAREA REȚELELOR

În prezent există numeroși producători și furnizori, fiecare cu propriile idei despre cum ar trebui realizate rețelele. În lipsa coordonării, ar fi un haos complet și utilizatorii nu ar putea face nimic. Singura soluție este să se convină asupra unor standarde de rețea.

Standardele nu numai că permit diverselor calculatoare să comunice între ele, ci sporesc totodată piața pentru produsele care aderă la un anumit standard, cu următoarele consecințe: producție de masă, profituri financiare, implementări VLSI și alte beneficii care duc la scăderea prețurilor și la acceptarea și mai largă a respectivelor produse. În secțiunile următoare vom arunca o privire asupra importantei, dar puțin cunoscutei, lumi a standardizării internaționale.

Standardele fac parte din două categorii: de facto și de jure. Standardele **de facto** (expresia latină pentru „de fapt") sunt acelea care pur și simplu au luat ființă, fără să existe vreun plan oficial. Deoarece zeci de producători au decis să copieze aproape identic mașinile IBM, PC-ul IBM și succesorii săi reprezintă standarde de facto pentru calculatoarele birourilor mici și pentru cele casnice. În secțiile de informatică ale facultăților, UNIX este standardul de facto pentru sisteme de operare.

Standardele **de jure** (expresia latină pentru „de drept") sunt, prin contrast, standarde legale, adoptate de un anumit organism de standardizare autorizat. Autoritățile de standardizare internaționale sunt, în general, împărțite în două clase: organizații stabilite prin tratate între guvernele naționale și organizații voluntare neguvernamentale. În domeniul standardelor pentru rețele de calculatoare există câteva organizații din fiecare categorie. În continuare vom discuta despre aceste organizații.

1.6.1 Who's Who în lumea telecomunicațiilor

Statutul legal al companiilor telefonice de pe glob variază considerabil de la țară la țară. La una din extreme se situează Statele Unite, care au 1500 de firme de telefonie private. Înainte să fie divizată, în 1984, AT&T, cea mai mare corporație din lume la vremea aceea, domina scena complet. AT&T furniza servicii telefonice pentru aproximativ 80 la sută din telefoanele Americii, răspândite pe jumătate din întinderea sa, în timp ce toate celelalte firme asigurau servicii pentru restul clienților (rurali, în majoritatea lor). De la divizarea sa, AT&T continuă să furnizeze servicii de lungă distanță, dar acum o face în concurență cu alte firme. Cele șapte Companii Regionale Bell în care a fost împărțit AT&T-ul, precum și alte numeroase firme independente, oferă servicii de telefonie locală și celulară. Datorită fuziunilor frecvente și a altor modificări de acest tip, această industrie este într-o continuă mișcare.

Firmele americane furnizoare de servicii de comunicații pentru public sunt numite **companii telefonice publice**. Ofertele și prețurile lor sunt descrise printr-un document numit **tarif**. Acesta trebuie să fie aprobat de Comisia Federală de Comunicații, care se ocupă de traficul dintre statele SUA și de traficul internațional, precum și de către comisiile publice de stat pentru traficul în interiorul său.

La cealaltă extremă se află țările în care guvernul are un monopol complet asupra tuturor mijloacelor de comunicație: poșta, telegraful, telefonul și, de multe ori, chiar radioul și televiziunea. Cea mai mare parte a lumii se încadrează în această categorie. În unele cazuri, autoritatea de telecomunicații este o companie naționalizată, în altele, este o simplă filială a guvernului, cunoscută de obicei sub numele de **PTT** (**Post, Telegraf & Telephone** administration). Tendința actuală în lumea întreagă este către liberalizare și competiție și împotriva monopolului guvernamental. Majoritatea țărilor europene și-au privatizat – mai mult sau mai puțin – sistemele PTT, dar peste tot acest proces este lent.

Din cauza tuturor acestor diverși furnizori de servicii este nevoie de o compatibilitate la scară mondială. Compatibilitatea asigură faptul că oamenii (și calculatoarele) dintr-o țară pot să-și apeleze partenerii din altă țară. La drept vorbind, această necesitate există de mult timp. În 1865, reprezentanți ai multor guverne din Europa s-au întâlnit pentru a forma predecesorul actualului **ITU** (**International Telecommunication Union**, rom: Uniunea Internațională de Telecomunicații). Sarcina Uniunii era standardizarea telecomunicațiilor internaționale, care la vremea aceea însemnau telegrafia. Chiar de atunci, era clar că dacă jumătate din țări foloseau codul Morse și cealaltă jumătate foloseau un cod diferit, atunci vor apare probleme. Când au apărut serviciile de telefonie internațională, ITU a preluat de asemenea și sarcina standardizării telefoniei (telephony – pronunțat și te-LEF-ony). În 1947 ITU a devenit o agenție a Națiunilor Unite. ITU are trei sectoare principale:

1. Sectorul de Radiocomunicații (ITU-R).
2. Sectorul de Standardizare a Telecomunicațiilor (ITU-T).
3. Sectorul de dezvoltare (ITU-D).

ITU-R se ocupă de alocarea frecvențelor internaționale de radio către grupurile concurente interesate. Ne vom referi mai întâi la ITU-T, care se ocupă de sistemele de telefonie și de comunicare de date. Din 1956 până în 1993, ITU-T a fost cunoscut ca **CCITT**, un acronim pentru numele său francez: Comité Consultatif International Télégraphique et Téléfonique. La 1 martie 1993, CCITT a fost reorganizat în scopul de a deveni mai puțin birocratic și a fost redenumit pentru a reflecta noul său rol. Atât ITU-T cât și CCITT au dat recomandări în domeniul telefoniei și comunicațiilor de date. Deși, începând cu 1993, recomandările poartă eticheta ITU-T, recomandările CCITT, de genul CCITT X.25, mai sunt încă frecvent întâlnite.

ITU-T are patru clase de membri:

1. Guverne naționale
2. Membri sectoriali
3. Membri asociați
4. Agenții de reglementare

ITU-T are aproximativ 200 de membri guvernamentali, incluzând aproape fiecare membru al Națiunilor Unite. Pentru că SUA nu are un sistem PTT, altcineva trebuia să o reprezinte în cadrul ITU-T. Această sarcină a revenit Departamentului de Stat, probabil pe principiul că ITU-T are de-a face cu țări străine, tocmai specialitatea acestui departament. Sunt aproximativ 500 de membri sectoriali, incluzând aici companiile de telefonie (AT&T, Vodafone, WorldCom), producătorii de echipamente de telecomunicații (Cisco, Nokia, Nortel), producătorii de echipamente de calcul (Compaq, Sun, Toshiba), producătorii de cipuri (Intel, Motorola, TI), companii media (AOL Time, Warner, CBS, Sony) și alte companii direct interesate (Boeing, Samsung, Xerox). Diverse organizații științifice non-profit, precum și consorții industriale sunt de asemenea membri sectoriali (IFIP, IATA). Membrii asociați sunt organizații mai mici care sunt interesate într-un anumit grup de studiu. Agențiile de reglementare sunt reprezentate de oamenii care supraveghează lumea afacerilor în telecomunicații, cum este de exemplu US Federal Communications Comission (Comisia Federală pentru Comunicații).

Sarcina pe care o are ITU-T este de a face recomandări tehnice asupra interfețelor din telefonie, telegrafie și comunicații de date. Acestea devin deseori standarde recunoscute internațional; de exemplu, V.24 (cunoscut în Statele Unite și ca EIA RS-232), specifică amplasarea și semnificația pinilor din conectorul folosit de majoritatea terminalelor asincrone și de modemurile externe.

Nu trebuie uitat că recomandările date de ITU-T sunt numai sugestii tehnice, pe care guvernele le pot adopta sau ignora, după cum doresc (pentru că guvernele sunt asemenea băieților de 13 ani – nu reacționează prea bine dacă li se dau ordine). În practică, o țară care dorește să adopte un standard de telefonie diferit de cel utilizat în restul lumii este liberă să o facă, dar o face cu prețul izolării de toate celelalte țări. Lucrul acesta poate să meargă în cazul Coreei de Nord, dar în altă parte ar fi o adevărată problemă. Fantezia de a numi standardele ITU-T „recomandări" a fost și este necesară pentru a calma forțele naționaliste din multe țări.

Adevărata muncă de la ITU-T se desfășoară în grupuri de studiu, care uneori cuprind chiar și 400 de persoane. Momentan sunt 14 grupuri de studiu, care acoperă subiecte de la facturarea serviciilor telefonice până la serviciile multimedia. Pentru ca până la urmă munca să aibă un rezultat, Grupurile de Studiu se împart în Echipe de Lucru, care se împart la rândul lor în Echipe de Experți, care, la rândul lor, se împart în grupuri ad-hoc. Birocrație a fost, birocrație rămâne.

În pofida tuturor acestor lucruri, ITU-T reușește să ducă la bun sfârșit ceea ce are de făcut. De la fondarea sa, a realizat mai bine de 3000 de recomandări, care ocupă peste 60.000 de pagini. Multe dintre acestea sunt folosite pe scară largă în practică. De exemplu, standardul V.90 56-Kbps pentru modemuri este o recomandare a ITU.

Pe măsură ce telecomunicațiile desăvârșesc tranziția - începută în anii 1980 - de la un caracter strict național la un caracter complet global, standardele vor deveni din ce în ce mai importante și tot mai multe organizații vor dori să devină implicate în producerea acestora. Pentru mai multe informații privind ITU, a se vedea (Irmer, 1994).

1.6.2 Who's Who în lumea standardelor internaționale

Standardele internaționale sunt produse de **ISO** (**International Standards Organization**[3], rom: Organizația Internațională de Standardizare), o organizație voluntară, neguvernamentală fondată în 1946. Membrii săi sunt organizațiile naționale de standardizare din cele 89 de țări membre. Acești membri cuprind ANSI (S.U.A.), BSI (Marea Britanie), AFNOR (Franța), DIN (Germania) și încă 85 de alte organizații.

ISO produce standarde referitoare la un număr vast de subiecte, începând cu piulițe și șuruburi și terminând cu vopsirea stâlpilor de telefon [pentru a nu menționa aici boabele de cacao (ISO 2451), plasele de pescuit (ISO 1530), lenjeria de damă (ISO 4416) și alte câteva subiecte la care nu v-ați putea gândi ca subiecte de standarde]. În total au fost create peste 5000 de standarde, inclusiv standardele OSI. ISO are aproape 200 de Comitete Tehnice (Technical Committees - TC), numerotate în ordinea creării lor, fiecare comitet ocupându-se de un subiect specific. TC1 se ocupă de piulițe și șuruburi (standardizarea înclinării filetelor). TC97 se ocupă de calculatoare și prelucrarea informației. Fiecare TC are subcomitete (SC-uri) împărțite în grupe de lucru (Work Groups - WG).

Munca propriu-zisă se desfășoară în principal în WG-uri, prin intermediul a peste 100.000 de voluntari din întreaga lume. Mulți dintre acești „voluntari" sunt puși să lucreze la probleme ale ISO de către patronii lor, ale căror produse sunt standardizate. Alții sunt oficiali guvernamentali dornici să vadă că modalitatea de a face lucrurile în țara lor devine standardul internațional. În multe WG-uri sunt activi, de asemenea, experți academici. În ceea ce privește standardele din telecomunicații, ISO și ITU-T cooperează frecvent, (ISO este un membru al ITU-T) în ideea de a evita ironia a două standarde internaționale oficiale și mutual incompatibile.

[3]Adevăratul nume pentru ISO este International Organization for Standardization (n.a.)

Număr	Subiect
802.1	Principiile generale si arhitectura LAN-urilor
802.2 ↓	Controlul legăturii logice
802.3 *	Ethernet
802.4 ↓	TokenBus (Jeton pe Magistrală – utilizat câtva timp în fabrici)
802.5	TokenRing (Jeton în Inel – contribuția IBM la lumea LAN)
802.6 ↓	Coadă dual, magistrală dual (rețea metropolitană timpurie)
802.7 ↓	Grupul de consiliere tehnică pe probleme de tehnologii de bandă largă
802.8 †	Grupul de consiliere tehnică pe probleme de tehnologii de fibră optică
802.9 ↓	LAN-uri izocrone pentru aplicații de timp real
802.10 ↓	LAN-uri virtuale şi securitate
802.11 *	LAN-uri fără fir
802.12 ↓	Prioritatea cererilor (AnyLAN de la HP)
802.13	Număr cu ghinion. Nimeni nu l-a vrut
802.14 ↓	Modemuri de cablu (decedat: un consorțiu industrial a abordat înainte domeniul)
802.15 *	Rețele personale (Bluetooth)
802.16 *	Comunicații fără fir în bandă largă
802.17	Inel activ de pachete

Fig. 1-38. Grupurile de lucru ale 802.
Cele importante sunt marcate cu *. Cele marcate cu ↓ hibernează.
Cele marcate cu † au renunțat și s-au desființat.

Reprezentantul S.U.A. în ISO este **ANSI** (**American National Standards Institute**, rom: Institutul Național American de Standarde), care, în pofida numelui său, este o organizație privată negu-vernamentală și nonprofit. Membrii săi sunt producători, companii telefonice publice și alte părți interesate. Standardele ANSI sunt frecvent adoptate de ISO ca standarde internaționale.

Procedura utilizată de ISO pentru adoptarea standardelor este concepută astfel încât să se obțină un consens cât mai larg posibil. Procesul începe când una din organizațiile naționale de standardizare simte nevoia unui standard internațional într-un anumit domeniu. În acel moment, se formează un grup de lucru care vine cu un **CD** (**Committee Draft**, rom: proiect de comitet). CD-ul circulă apoi pe la toate organizațiile membre, care au la dispoziție 6 luni pentru a-l supune criticilor. Dacă se primește aprobarea din partea unei majorități substanțiale, atunci se produce un document revizuit, numit **DIS** (**Draft International Standard**, rom: proiect de standard internațional), care va circula în scopul de a fi comentat și votat. Pe baza rezultatelor din această rundă, se pregătește, se aprobă și se publică textul final al respectivului **IS** (**International Standard**, rom: standard internațional). În domeniile foarte controversate, un CD sau un DIS pot să treacă prin câteva versiuni înainte de a obține suficiente voturi și întregul proces poate dura ani de zile.

NIST (**National Institute of Standards and Technology**, rom: Institutul Național de Standarde și Tehnologie) este o agenție a Departamentului pentru Comerț al Statelor Unite. NIST a fost cunoscut anterior sub numele de Biroul Național de Standarde. El produce standarde care sunt obligatorii pentru achizițiile făcute de guvernul U.S.A., mai puțin pentru cele care privesc Departamentul de Apărare, acesta având propriile sale standarde.

Un alt actor important din lumea standardelor este **IEEE** (**Institute of Electrical and Electronics Engineers**, rom: Institutul Inginerilor Electricieni și Electroniști), cea mai mare organizație profesională din lume. Suplimentar față de producerea a zeci de jurnale și organizarea a numeroase conferințe în fiecare an, IEEE are un grup de standardizare care dezvoltă standarde în domeniul ingineriei electrice și tehnicii de calcul. Comitetul IEEE 802 a standardizat mai multe tipuri de rețele locale. Vom studia o parte dintre rezultatele sale ceva mai târziu în această carte. Munca efectivă este făcută de o sumă de grupuri de lucru, care sunt prezentate în fig. 1-38. Rata de succes a diverselor grupuri ale 802 a fost scăzută, așadar chiar dacă ai un număr de forma 802.x, aceasta nu este o garanție a succesului. Dar impactul poveștilor de succes (în special 802.3 și 802.11) a fost enorm.

1.6.3 Who's Who în lumea standardelor Internet

Internet-ul mondial are propriile sale mecanisme de standardizare, foarte diferite de cele ale ITU-T și ISO. Diferența poate fi rezumată grosier spunând că lumea care vine la întâlnirile pentru standardizare ale ITU și ISO poartă costum. Lumea care vine la întâlnirile pentru standardizarea Internet-ului poartă blugi (iar dacă se întâlnesc la San Diego poartă pantaloni scurți și tricouri).

La întâlnirile organizate de ITU-T și ISO e plin de oficiali ai unor corporații și de funcționari guvernamentali pentru care standardizarea reprezintă meseria lor. Ei privesc standardizarea ca un lucru bun și își dedică viețile acestui scop. Lumea implicată în Internet, pe de altă parte, preferă, ca principiu de bază, anarhia. Oricum, dacă sute de milioane de oameni își văd fiecare numai de treaba lor, este puțin probabil să apară vreo modalitate de comunicare. De aceea, standardele, deși regretabile, apar ocazional ca fiind necesare.

Când a fost creat ARPANET-ul, DoD-ul a înființat un comitet neoficial care să îl supravegheze. În 1983 comitetul a fost redenumit **IAB** (**Internet Activities Board**, rom: Consiliul Activităților Internet) și a primit o misiune ceva mai amplă: să fie atent ca cercetătorii implicați în ARPANET și Internet să se miște, mai mult sau mai puțin, în aceeași direcție - o activitate care ar putea fi asemănată cu „păstoritul" pisicilor. Semnificația acronimului „IAB" a fost schimbată mai târziu în **Internet Architecture Board** (Consiliul Arhitecturii Internet).

Fiecare din cei aproximativ 10 membri ai IAB-ului conducea un departament care se ocupa de o anumită problemă importantă. IAB-ul se întâlnea de câteva ori pe an pentru a discuta rezultatele și a trimite informări către DoD și NSF, care asigurau la acea vreme majoritatea fondurilor. Când era nevoie de un nou standard (de exemplu, un nou algoritm de dirijare), membrii IAB îl luau în discuție și apoi anunțau schimbarea, astfel ca absolvenții facultăților - care erau sufletul muncii de programare - să îl poată implementa. Comunicările erau puse la dispoziție printr-o serie de rapoarte tehnice, numite **RFC**-uri (**Request For Comments**, rom: cereri pentru comentarii). RFC-urile sunt memorate on-line și pot fi citite de oricine este interesat de ele la adresa *www.ietf.org/rfc*. RFC-urile sunt numerotate în ordinea cronologică a creării lor. Până acum există peste 3000. Ne vom referi la multe dintre ele în cursul acestei cărți.

În 1989 Internet-ul crescuse atât de mult, încât acest stil informal nu mai putea funcționa. Multe firme vindeau la acea vreme produse TCP/IP și nu erau dispuse să le modifice doar pentru că zece cer-

cetători se gândiseră la o idee mai bună. În vara anului 1989, IAB a fost reorganizat. Cercetătorii au fost transferați la **IRTF** (**Internet Research Task Force**, rom: Departamentul de Cercetare Internet), care a fost pus în subordinea IAB-ului, alături de **IETF** (**Internet Engineering Task Force**, rom: Departamentul de Inginerie Internet). IAB-ul a fost repopulat cu persoane care reprezentau un palier de organizații mai larg decât stricta comunitate a cercetătorilor. La început a fost un grup care se autoperpetua: membrii erau activi pe o perioadă de 2 ani, iar membrii noi erau selectați de către membrii mai vechi. Mai târziu, a fost înființată **Societatea Internet** (**Internet Society**), care reunea oameni interesați de Internet. Societatea Internet este, prin urmare, comparabilă într-un sens cu ACM sau IEEE. Societatea este administrată de un comitet ales, iar comitetul desemnează membrii IAB.

Ideea acestei divizări a fost ca IRTF să se concentreze asupra cercetării pe termen lung, iar IETF să se ocupe de probleme inginerești pe termen scurt. IETF a fost împărțit în grupuri de lucru, fiecare cu o problemă specifică de rezolvat. Inițial, președinții grupurilor de lucru s-au reunit într-un comitet de organizare, în scopul de a coordona munca inginerească ce le revenea. Preocupările grupurilor de lucru includeau aplicații noi, informații de la utilizatori, integrare OSI, dirijare și adresare, securitate, administrare de rețea, standarde. În final s-au format atât de multe grupuri de lucru (mai mult de 70), încât ele au fost grupate pe domenii, iar comitetul de organizare s-a constituit din președinții domeniilor.

În plus, a fost adoptat un proces de standardizare mai formal, preluat după modelul ISO. Pentru a deveni un standard propus (**Proposed Standard**), ideea fundamentală trebuie să fie complet explicată într-un RFC și să prezinte destul interes din partea comunității pentru a merita să fie luată în considerare. Pentru a avansa la stadiul de proiect de standard (**Draft Standard**), este necesară o implementare de lucru care să fi fost testată în amănunțime de către două situri independente, timp de cel puțin 4 luni. Dacă IAB-ul este convins că ideea e bună și că programul funcționează, atunci poate să declare RFC-ul respectiv ca fiind un Standard Internet. Unele Standarde Internet au devenit standarde ale DoD-ului (MIL-STD), fiind, prin urmare, obligatorii pentru furnizorii DoD-ului. David Clark a făcut odată o remarcă devenită celebră privitoare la standardizarea Internet-ului, care ar consta din „consens aproximativ și programe care merg."

1.7 UNITĂȚI DE MĂSURĂ

Pentru a ne feri de orice confuzie, merită să precizăm de la bun început că în această carte, ca și în lumea științei calculatoarelor în general, vor fi folosite unitățile metrice în locul unităților tradiționale englezești (sistemul furlong-stone-fortnight[4]). Principalele prefixe metrice sunt precizate în fig. 1-39. Ale sunt în general abreviate folosindu-se prima literă, cu unitățile mai mari ca 1 scrise cu majuscule (KB, MB etc.). O excepție (din motive istorice) este Kbps (kilobits per second) pentru kilobiți pe secundă. Astfel, o linie de comunicație de 1 Mbps transmite 10^6 biți/secundă, în timp ce pentru 100 ps (psec), ceasul bate la fiecare 10^{-10} secunde. Deoarece denumirile mili și micro încep amândouă cu litera „m", trebuia făcută o alegere. În mod normal, „m" este folosit pentru mili, iar „μ" (litera greacă *miu*) este folosit pentru micro.

[4] furlong = jumătate de milă
stone = 6,350kg
fortnight = 2 săptămâni

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Fig. 1-39. Principalele prefixe metrice

Este de asemenea important să subliniem că pentru măsurarea dimensiunilor memoriei, discurilor, fișierelor și a bazelor de date se obișnuiește folosirea acestor unități, deși ele au valori ușor modificate. Astfel, kilo reprezintă 2^{10} (1024) și nu de 10^{3} (1000), pentru că volumului memoriilor sunt întotdeauna puteri ale lui doi. Deci, o memorie de 1 KB are 1024 de octeți, nu 1000. Similar, o memorie de 1 MB are 2^{20} (1.048.576) octeți, o memorie de 1 GB are 2^{30} octeți (1.073.741.824), iar o bază de date de 1 TB are 2^{40} (1.099.511.627.776) octeți. Oricum, o linie de comunicație de 1 Kbps transmite 1000 de biți pe secundă și o rețea locală de 10 Mbps rulează la 10.000.000 biți/secundă, deoarece aceste unități nu sunt puteri ale lui 2. Din păcate, mulți oameni tind să amestece aceste două sisteme, în special pentru capacitatea discurilor. Pentru a evita orice ambiguitate, în această carte vom folosi simbolurile KB, MB, GB pentru 2^{10}, 2^{20}, 2^{30}, și simbolurile Kbps, Mbps și Gbps pentru 10^{3}, 10^{6} și 10^{9} biți pe secundă, respectiv.

1.8 RESTUL CĂRȚII ÎN REZUMAT

Cartea de față discută atât principiile cât și practica interconectării calculatoarelor. Majoritatea capitolelor încep printr-o discuție a principiilor relevante, urmată de un număr de exemple care ilustrează principiile respective. Aceste exemple sunt în general preluate din Internet și din rețele fără fir deoarece acestea sunt importante și diferite. Acolo unde este relevant, vor fi date și alte exemple.

Cartea este structurată în concordanță cu modelul hibrid din fig. 1-24. Începând cu cap. 2, pornim la drum de la bază în sus, de-a lungul ierarhiei de protocoale. Cap. 2 prezintă cadrul pentru studierea domeniului comunicațiilor de date. Capitolul acoperă diferite subiecte: transmisii analogice și digitale, multiplexare, comutare, sistemul telefonic trecut, actual și viitor. Acoperă sisteme de transmisie cu cablu, fără cablu și prin satelit. Acest material se referă la nivelul fizic, dar noi ne vom ocupa numai de aspectele arhitecturale, nu de cele privitoare la echipamente. Sunt discutate, de asemenea, câteva exemple de niveluri fizice, cum ar fi rețeaua cu comutare a telefoniei publice, telefoanele mobile și televiziunea prin cablu.

Cap. 3 discută modelul legătură de date și protocoalele sale prin intermediul unui număr de exemple din ce în ce mai complexe. Se realizează, de asemenea, analiza acestor protocoale. După aceea, sunt discutate unele protocoale importante din lumea reală, printre care HDLC (folosit în rețelele de viteză scăzută și medie) și PPP (folosit în Internet).

Cap. 4 se referă la subnivelul de acces la mediu, care face parte din nivelul legătură de date. Problema fundamentală cu care se ocupă este cum să determine cine poate folosi rețeaua - atunci când rețeaua constă dintr-un singur canal partajat, așa cum se întâmplă în majoritatea LAN-urilor și în unele rețele de sateliți. Sunt date multe exemple din domeniul LAN-urilor cu cablu sau fără (în special Ethernet), din cel al MAN-urilor fără fir, din cadrul rețelelor bazate pe Bluetooth și al rețelelor de sateliți. Tot aici sunt discutate și punțile, care se folosesc pentru a interconecta LAN-urile.

Cap. 5 se ocupă de nivelul rețea, în special de dirijare, cu prezentarea mai multor algoritmi de dirijare, atât statici cât și dinamici. Chiar dacă se folosesc algoritmi de rutare foarte buni, dacă traficul cerut este mai mare decât cel pe care îl poate dirija rețeaua, se ajunge la congestia rețelei, așa că se va discuta despre congestie și despre cum poate fi ea evitată. O variantă încă și mai bună decât evitarea congestiei este oferirea unei garanții de calitate a serviciilor. Și acest subiect va fi abordat aici. Interconectarea rețelelor eterogene în inter-rețele conduce la numeroase probleme care sunt discutate aici. Se acordă mare atenție nivelurilor din Internet .

Cap. 6 se ocupă de nivelul transport. Se discută pe larg protocoalele orientate pe conexiuni, deoarece ele sunt necesare în numeroase aplicații. Se discută în detaliu un exemplu de serviciu de transport și implementarea sa. Este prezentat chiar și codul sursă pentru acest exemplu simplu, pentru a se putea demonstra modul în care poate fi el implementat. Ambele protocoale din Internet – UDP și TCP – sunt discutate în detaliu și este abordată problema performanțelor lor. În plus, se discută despre problemele impuse de rețelele fără fir.

Cap. 7 se ocupă de nivelul aplicație, de protocoalele și aplicațiile sale. Primul subiect este DNS, care este cartea de telefoane a Internet-ului. Apoi urmează poșta electronică, inclusiv o discuție despre protocoalele sale. Apoi ne vom muta atenția asupra Web-ului, cu discuții detaliate despre conținut static, conținut dinamic, ce se întâmplă la client, ce se întâmplă pe server, protocoale, performanță, Web fără fir. În cele din urmă vom examina informația multimedia care este transmisă prin rețea, inclusiv fluxuri audio, radio prin Internet și video la cerere.

Cap. 8 se referă la securitatea rețelelor. Acest subiect include aspecte legate de fiecare dintre niveluri, așa că este mai ușor de tratat către final, când toate nivelurile au fost deja explicate pe larg. Capitolul începe cu o introducere în criptografie. În continuare, este prezentat modul în care criptografia poate fi utilizată pentru a securiza comunicațiilor, poșta electronică și Web-ul. Cartea se încheie cu o discuție despre anumite domenii în care securitatea interferează cu intimitatea, libertatea de exprimare, cenzura, precum și alte probleme sociale care decurg de aici.

Cap. 9 conține o listă adnotată de lecturi sugerate, aranjate în ordinea capitolelor. Lista este gândită ca un ajutor pentru cititorii care doresc să continue studiul rețelelor. Capitolul are de asemenea o bibliografie alfabetică a tuturor referințelor citate în această carte.

Situl Web al autorului de la Prentice Hall: *http://www.prenhall.com/tanenbaum* are o pagină cu legături la mai multe sinteze, liste de întrebări frecvente (FAQs), companii, consorții industriale, organizații profesionale, organizații de standardizare, tehnologii, lucrări științifice și altele.

1.9 REZUMAT

Rețelele de calculatoare pot fi utilizate pentru numeroase servicii, atât pentru firme cât și pentru persoane particulare. Pentru companii, rețelele de calculatoare personale care folosesc servere par-

tajate asigură accesul la informațiile corporației. De obicei, acestea urmează modelul client-server, cu stațiile de lucru clienți pe mesele de lucru ale angajaților accesând serverele puternice din camera mașinilor. Pentru persoane particulare, rețelele oferă acces la o mulțime de informații și de resurse de divertisment. De cele mai multe ori persoanele particulare accesează Internet-ul folosind un modem pentru a apela un ISP, deși din ce în ce mai mulți utilizatori au chiar și acasă o conexiune Internet fixă, permanentă. Un domeniu care se dezvoltă rapid este acela al rețelelor fără fir, care conduc la dezvoltarea de noi aplicații, cum ar fi mobilitatea accesului la poșta electronică și comerțul mobil.

În mare, rețelele pot fi împărțite în LAN-uri, MAN-uri, WAN-uri și inter-rețele, fiecare cu caracteristicile, tehnologiile, vitezele și rolurile sale proprii. LAN-urile acoperă suprafața unei clădiri și lucrează la viteze mari, MAN-urile acoperă suprafața unui oraș – de exemplu rețeaua de televiziune prin cablu, care este actualmente folosită de mulți dintre utilizatori și pentru conectarea la Internet. WAN-urile se întind pe suprafața unei țări sau a unui continent. LAN-urile și MAN-urile sunt necomutate (adică nu au rutere); WAN-urile sunt comutate. Rețelele fără fir devin din ce în ce mai populare, în special la nivelul rețelelor locale. Rețelele pot fi interconectate pentru a forma inter-rețele.

Programele de rețea constau din protocoale, adică reguli prin care procesele pot să comunice. Protocoalele pot fi fie fără conexiuni, fie orientate pe conexiuni. Majoritatea rețelelor asigură suport pentru ierarhiile de protocoale, fiecare nivel asigurând servicii pentru nivelurile de deasupra sa și izolându-le de detaliile protocoalelor folosite în nivelurile de mai jos. Stivele de protocoale se bazează în mod tipic fie pe modelul OSI, fie pe modelul TCP/IP. Ambele modele posedă niveluri rețea, transport și aplicație, dar ele diferă în ceea ce privește celelalte niveluri. Problemele care apar în procesul de proiectare a acestor protocoale includ multiplexarea, controlul traficului, controlul erorilor și încă altele. O mare parte a acestei cărți este dedicată protocoalelor și proiectării lor.

Rețelele oferă servicii utilizatorilor lor. Aceste servicii pot fi orientate pe conexiune sau fără conexiune. În anumite rețele, serviciile fără conectare sunt oferite la un anumit nivel și pot fi completate cu serviciile orientate pe conexiune oferite de un alt nivel.

Ca rețele bine-cunoscute sunt menționate Internet-ul, rețelele ATM, Ethernet-ul și LAN-ul fără fir, standard denumit IEEE 802.11. Internet-ul a evoluat din ARPANET, prin adăugarea de noi rețele pentru a se forma o inter-rețea. În prezent, Internet-ul este în fapt o colecție de multe mii de rețele și nu o singură rețea. Ceea ce caracterizează această colecție este folosirea stivei TCP/IP peste tot. Rețelele ATM sunt răspândite mai ales în sistemele de telefonie pentru trafic de date intensiv. Ethernet-ul este cea mai populară rețea locală și este implementată în majoritatea companiilor mari și în universități. În fine, rețelele locale fără fir, cu viteze de transfer surprinzător de mari (până la 54 Mbps) încep să fie folosite pe scară largă.

Pentru a putea determina mai multe calculatoare să comunice între ele este nevoie de o importantă muncă de standardizare, atât pentru partea de echipamente (hardware), cât și pentru partea de programe (software). Organizațiile ca ITU-T, ISO, IEEE și IAB administrează diverse părți din procesul de standardizare.

1.10 PROBLEME

1. Imaginați-vă că v-ați dresat câinele St. Bernard, pe nume Bernie, ca, în locul clasicei sticle cu rom, să poarte o cutie cu trei benzi de 8 mm. (Când ți se umple discul, respectiva cutie reprezintă o ur-

gență.) Aceste benzi conțin fiecare câte 7 gigabytes. Câinele poate călători până la dvs., oriunde v-ați afla, cu 18 km/h. Pentru ce ordin de distanțe are Bernie o viteză mai mare de transmisie a datelor decât o linie a cărei viteză de transfer (fără supraîncărcare) este de 150 Mbps?

2. O alternativă la un LAN este pur și simplu un mare sistem, cu divizarea timpului cu terminale pentru toți utilizatorii. Prezentați două avantaje ale unui sistem client-server care folosește un LAN.

3. Performanța unui sistem client-server este influențată de doi factori ai rețelei: lărgimea de bandă (câți biți poate transporta într-o secundă) și latența (câte secunde durează transferul primului bit de la client la server). Dați un exemplu de rețea care are și lărgime de bandă ridicată și latență mare. Apoi dați un exemplu de rețea cu lărgime de bandă scăzută și latență mică.

4. Pe lângă lărgime de bandă și latență, ce alt parametru este necesar pentru a caracteriza calitatea serviciilor oferite de o rețea folosită pentru trafic de voce digitizată?

5. Un factor de întârziere al unui sistem memorează-și-retransmite cu comutare de pachete este cât de mult timp ia operația de stocare și retrimitere a unui mesaj printr-un comutator. Dacă timpul de comutare este de 10 μs, este acesta un factor important în răspunsul unui sistem client-server în care clientul este în New York și serverul în California? Presupuneți că viteza de propagare a semnalului printr-un fir de cupru sau prin fibra optică ar fi de 2/3 din viteza luminii în vid.

6. Un sistem client-server folosește o rețea-satelit, cu satelitul amplasat la o înălțime de 40.000 km. În cazul optim, care este întârzierea cu care vine răspunsul la o cerere?

7. În viitor, când toată lumea va avea acasă un terminal conectat la o rețea de calculatoare, vor deveni posibile referendumuri publice imediate pe subiecte de legislație importante. În ultimă instanță ar putea fi chiar eliminate parlamentele, pentru a lăsa voința poporului să se exprime direct. Aspectele pozitive ale unei astfel de democrații directe sunt destul de evidente; discutați unele din aspectele negative.

8. O colecție de cinci rutere trebuie să fie conectată într-o subrețea punct-la-punct. Între două rutere proiectanții pot instala o linie de mare viteză, o linie de viteză medie, o linie de viteză scăzută sau nici o linie. Dacă generarea și examinarea fiecărei topologii pe calculator durează 100 ms, cât timp va dura examinarea tuturor topologiilor pentru a o găsi pe cea care se potrivește cel mai bine cu încărcarea prevăzută?

9. Un grup de 2^n-1 rutere sunt interconectate într-un arbore binar centralizat, cu un ruter în fiecare nod al arborelui. Ruterul *i* comunică cu ruterul *j* trimițând un mesaj rădăcinii arborelui. Rădăcina trimite apoi mesajul înapoi în jos până la *j*. Deduceți o expresie aproximativă pentru numărul mediu de salturi pe mesaj în cazul unui număr *n* mare, presupunând că toate perechile de rutere sunt la fel de probabile.

10. Un dezavantaj al unei subrețele cu difuzare este risipa de capacitate datorată multiplelor gazde care încearcă să acceseze canalul în același timp. Ca un exemplu simplist, să presupunem că timpul este împărțit în intervale discrete și fiecare din cele *n* gazde încearcă să utilizeze canalul cu probabilitatea *p* în timpul fiecărui interval. Ce fracțiune din intervale se pierde datorită coliziunilor?

11. Care sunt două din motivele utilizării protocoalelor organizate pe niveluri?

12. Președintelui Companiei de Vopsele Speciale îi vine ideea să lucreze împreună cu un producător local de bere în scopul de a produce o cutie de bere invizibilă (ca o măsură anti-gunoi). Președintele comandă departamentului său juridic să analizeze ideea, iar acesta cere ajutorul, la rândul său, departamentului de ingineri. Ca rezultat, inginerul șef îl cheamă pe inginerul-șef de la cealaltă firmă pentru a discuta aspectele tehnice ale proiectului. Apoi, inginerii prezintă un raport către departamentele juridice respective, iar acestea aranjează prin telefon aspectele legale. În final, cei doi președinți de firme discută partea financiară a afacerii. Este acesta un exemplu de protocol multinivel în sensul modelului OSI?**6.** Care sunt adresele SAP în cazul difuzării radio FM ?

13. Care este principala diferență între comunicarea fără conexiuni și comunicarea orientată pe conexiuni?

14. Două rețele furnizează, fiecare, servicii orientate pe conexiuni sigure. Una din ele oferă un flux sigur de octeți, iar cealaltă oferă un flux sigur de mesaje. Sunt acestea identice? Dacă da, de ce se face această distincție? Dacă nu, exemplificați prin ce diferă.

15. Ce înseamnă „negociere” atunci când se discută protocoalele de rețea? Dați un exemplu.

16. În fig. 1-19 este prezentat un serviciu. Există și servicii implicite în această figură? Dacă da, unde? Dacă nu, de ce nu?

17. În unele rețele, nivelul legătură de date tratează erorile de transmisie, solicitând retransmiterea cadrelor deteriorate. Dacă probabilitatea de a se strica un cadru este p, care este numărul mediu de transmisii necesare pentru a trimite un cadru, în cazul în care confirmările nu se pierd niciodată?

18. Care dintre nivelurile OSI se ocupă de fiecare din următoarele sarcini:
a) Descompunerea fluxului de biți transmiși în cadre.
b) Determinarea traseului care trebuie folosit în subrețea.
c) TDPU-rile încapsulează pachete sau invers? Discuție.

19. Dacă unitățile de date schimbate la nivelul legătură de date se numesc cadre și unitățile de date schimbate la nivelul rețea se numesc pachete, pachetele încapsulează cadre sau cadrele încapsulează pachete? Explicați răspunsul dat.

20. Un sistem are o ierarhie de protocoale organizată pe n niveluri. Aplicațiile generează mesaje de lungime M octeți. La fiecare nivel este adăugat un antet de h octeți. Ce fracțiune din lățimea benzii rețelei este ocupată de antete?

21. Prezentați două aspecte comune modelului de referință OSI și modelului de referință TCP/IP. Prezentați apoi două aspecte prin care modelele diferă.

22. Care este principala deosebire între TCP și UDP?

23. Subrețeaua din fig. 1-25(b) a fost proiectată pentru a putea rezista unui război nuclear. Câte bombe ar fi necesare pentru a partiționa nodurile sale în două seturi complet deconectate? Presupuneți că orice bombă distruge un nod și toate legăturile conectate cu el.

24. Internet-ul își dublează dimensiunea o dată la aproximativ 18 luni. Deși nimeni nu știe cu siguranță, se estimează numărul gazdelor la 100 de milioane în 2001. Folosiți aceste date pentru a calcula numărul de gazde Internet prevăzut pentru anul 2010. Puteți crede acest scenariu? Explicați de ce da sau de ce nu.

25. La transferul unui fișier între două calculatoare există (cel puțin) două strategii de confirmare. Conform primei strategii, fișierul este descompus în pachete care sunt confirmate individual de către server, dar transferul de fișiere pe ansamblu nu este confirmat. În a doua strategie, pachetele nu sunt confirmate individual, dar la sfârșit este confirmat întregul fișier. Discutați aceste două abordări.

26. De ce folosește ATM-ul celule mici, de lungime fixă?

27. Cât de lung era un bit în standardul original 802.3 măsurat în metri? Folosiți viteza de transmisie de 10 Mbps și presupuneți că viteza de transmisie prin cablu coaxial este de 2/3 din viteza de propagare a luminii în vid.

28. O imagine are 1024 x 768 pixeli și reține câte 3 octeți pentru fiecare pixel. Presupuneți că imaginea este necomprimată. Cât durează transmisia ei pe un canal de modem de 56 Kbps ? Dar printr-un modem de cablu de 1 Mbps? Dar prin Ethernet la 10 Mbps? Dar prin Ethernet la 100 Mbps ?

29. Ethernet-ul și rețelele fără fir au unele asemănări și deosebiri. O proprietate a Ethernet-ului este aceea că un singur cadru poate fi transmis la un moment dat pe mediu. Are și 802.11 această proprietate? Discutați răspunsul dat.

30. Rețelele fără fir sunt ușor de instalat, ceea ce le face mai ieftine, deoarece de cele mai mult ori operația de instalare depășește semnificativ costul echipamentelor. Totuși, aceste rețele au și unele dezavantaje. Numiți două dintre ele.

31. Prezentați două avantaje și două dezavantaje ale existenței standardelor internaționale pentru protocoalele de rețea.

32. Atunci când un sistem dispune de o parte permanentă și de o parte detașabilă, de exemplu un cititor de CD-uri și un CD-ROM, este important ca sistemul să fie standardizat, astfel ca diferite firme să poată realiza atât părțile permanente cât și cele mobile și ca ele să se potrivească fără probleme. Dați trei exemple din afara industriei de calculatoare unde există astfel de standarde internaționale. Indicați apoi trei domenii din afara industriei de calculatoare unde nu există astfel de standarde.

33. Alcătuiți o listă de activități pe care le faceți zilnic și în care sunt implicate rețele de calculatoare. Cum ar fi viața voastră alterată dacă aceste rețele ar fi deconectate la un moment dat ?

34. Descoperiți ce rețele sunt utilizate în școala sau la locul de muncă. Descrieți tipurile de rețele, topologiile și metodele de comutare folosite acolo.

35. Programul *ping* vă permite să trimiteți un pachet de test la o locație dată pentru a vedea cât de mult durează până când acesta ajunge acolo și înapoi. Încercați să folosiți *ping* pentru a vedea cât de mult durează transferul pachetului între locul în care vă găsiți și alte câteva locuri cunos-

cute. Din aceste date, calculați timpul de tranzit într-o sigură direcție în funcție de distanță. Este bine să folosiți universitățile deoarece locațiile serverelor lor sunt cunoscute foarte bine. De exemplu, *berkley.edu* este în Berkley, California, *mit.edu* este în Cambridge, Massachusetts, *vu.nl* este în Amsterdam, Olanda, *www.usyd.edu.au* este în Sydney, Australia și *www.uct.ac.za* este în Cape Town, Africa de Sud.

36. Vizitați situl Web al IETF, *www.ietf.org* pentru a vedea ce mai fac. Alegeți un proiect care vă place și scrieți un raport de jumătate de pagină despre problemă și despre o soluție propusă.

37. Standardizarea este foarte importantă în lumea rețelelor. ITU și ISO sunt principalele organizații oficiale de standardizare. Vizitați siturile lor Web, *www.itu.org* și *www.iso.org*, respectiv, și aflați despre munca lor de standardizare. Scrieți un scurt raport despre tipurile de lucruri pe care le-au standardizat.

38. Internet-ul este alcătuit dintr-un mare număr de rețele. Aranjarea lor determină topologia Internet-ului. O importantă cantitate de informații despre topologia Internet-ului este disponibilă online. Folosiți un motor de căutare pentru a afla mai multe despre acest subiect și scrieți un scurt raport care să rezume informațiile pe care le-ați găsit.

2

NIVELUL FIZIC

În continuare vom analiza trei tipuri de medii de transmisie: ghidate (cablu din cupru și fibre optice), fără fir (unde radio terestre) și prin satelit. Acest material furnizează informațiile fundamentale referitoare la tehnologiile de comunicație folosite în rețelele moderne.

Restul capitolului este dedicat descrierii a trei exemple de sisteme de comunicație folosite în practică pentru rețele cu răspândire geografică largă. Vom începe cu sistemul telefonic, studiind trei variante: sistemul de telefonie fixă, sistemul de telefonie mobilă și sistemul bazat pe cablu de televiziune. Toate acestea folosesc fibra optică pentru implementarea coloanei vertebrale, dar sunt organizate diferit și folosesc tehnologii diferite pentru ultima milă a legăturii.

2.1 BAZELE TEORETICE ALE COMUNICĂRII DE DATE

Informația poate fi transmisă prin cablu folosind variația unor proprietăți fizice ale semnalului cum ar fi tensiunea și intensitatea curentului. Reprezentând valoarea tensiunii sau a intensității curentului ca o funcție de timp, *f(t)*, putem modela comportamentul semnalului și îl putem analiza matematic. Această analiză face subiectul următoarelor secțiuni.

2.1.1 Analiza Fourier

La începutul secolului XIX, matematicianul francez Jean-Baptiste Fourier a demonstrat că orice funcție *g(t)*, cu evoluție rezonabilă și periodică cu perioada T, poate fi construită prin însumarea unui număr (posibil infinit) de sinusoide și cosinusoide:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \qquad (2\text{-}1)$$

unde $f = 1 / T$ este frecvența fundamentală, iar a_n și b_n sunt amplitudinile sinusoidelor și cosinusoidelor **armonicei** (termenului) de ordinul n, iar *c* este o constantă. Această descompunere este numită **serie Fourier**. Pornind de la seria Fourier, funcția poate fi reconstruită; aceasta înseamnă că, dacă perioada *T* este cunoscută și amplitudinile sunt date, funcția de timp originală poate fi obținută prin evaluarea sumelor din ecuația 2-1.

Un semnal de durată finită (proprietate pe care o au toate semnalele) poate fi tratat presupunându-se că el repetă un anumit tipar la infinit (de exemplu, semnalul este același în intervalul de la *T* la *2T* ca în intervalul de la 0 la *T*, etc.).

Amplitudinile a_n pot fi calculate pentru orice *g(t)* dat prin multiplicarea ambilor membri ai ecuației 2-1 cu $\sin(2\pi kft)$ urmată de integrarea de la 0 la *T*. Deoarece

$$\int_0^T \sin(2\pi kft)\sin(2\pi nft)dt = \begin{cases} 0 & \text{pentru } k \neq n \\ T/2 & \text{pentru } k = n \end{cases}$$

numai un singur termen al sumei nu se anulează: a_n . Suma de cosinusuri – cea cu b_n – se anulează complet. Similar, multiplicând membrii ecuației 2-1 cu $\cos(2\pi kft)$ și integrând de la 0 la *T*, putem obține b_n. Prin integrarea ambilor membri ai ecuației originale, se poate obține *c*. Rezultatele obținute prin efectuarea acestor operații sunt:

$$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi nft)dt \quad b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)dt \quad c = \frac{2}{T}\int_0^T g(t)dt$$

2.1.2 Semnalele cu bandă de frecvență limitată

Pentru a face legătura dintre cele prezentate și comunicația de date să considerăm următorul exemplu: transmisia caracterului ASCII „b" codificat pe un octet. Biții care urmează a fi transmiși sunt 01100010. Partea din stânga a fig. 2-1(a) reprezintă tensiunea la ieșire emisă de calculatorul care transmite. Din analiza Fourier a acestui semnal rezultă următorii coeficienți:

$$a_n = \frac{1}{\pi n}[\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$b_n = \frac{1}{\pi n}[\sin(3\pi n/4) - \sin(\pi n/4) + \sin(7\pi n/4) - \sin(6\pi n/4)]$$

$$c = 3/4$$

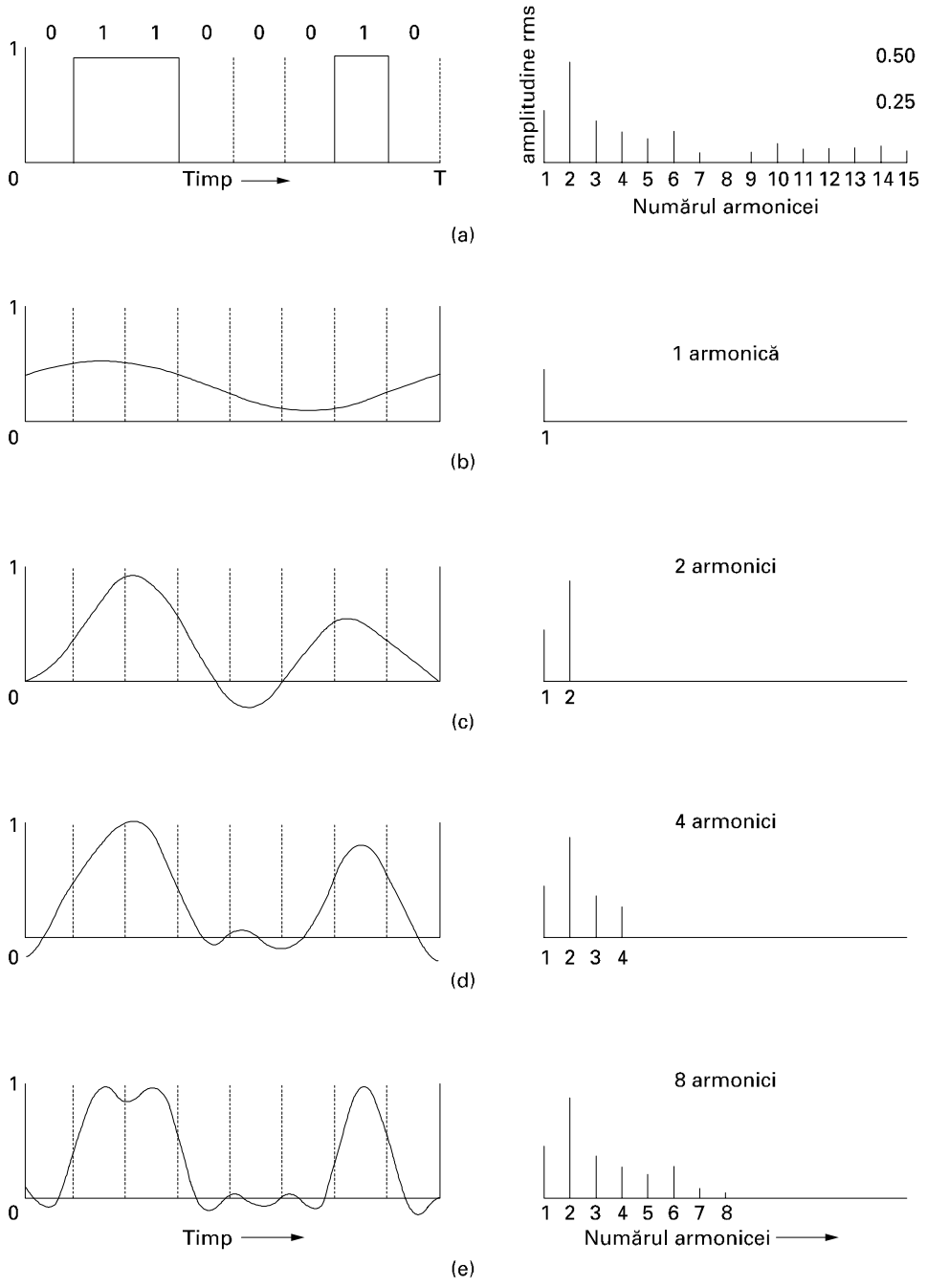


Fig. 2-1. (a) Un semnal binar și radicalul sumei pătratelor amplitudinilor Fourier. (b) - (e) Aproximații succesive ale semnalului inițial.

Radicalul sumei pătratelor amplitudinilor, $\sqrt{a_n^2 + b_n^2}$, pentru primii termeni este prezentat în partea dreaptă a fig. 2-1(a). Aceste valori sunt cele care ne interesează, deoarece pătratele lor sunt proporționale cu energia transmisă la frecvența respectivă.

Nu există un mijloc de transmisie care să poată trimite semnale fără pierdere de putere în timpul procesului. Dacă toate componentele Fourier ar fi micșorate în aceeași măsură, atunci semnalul rezultat ar fi atenuat în amplitudine, dar nu ar prezenta distorsiuni [ar avea aceeași formă ca cea din fig. 2-1(a)]. Din păcate, orice mijloc de transmisie atenuează componente Fourier diferite cu factori diferiți, introducând astfel distorsiuni. De obicei, amplitudinile sunt transmise fără atenuări de la 0 la o anumită frecvență f_p [măsurată în cicluri/secundă sau în Hertzi (Hz)] și toate celelalte componente cu frecvențe mai mari decât această frecvență de tăiere sunt puternic atenuate. Intervalul de frecvențe transmise fără a fi atenuate semnificativ se numește **lărgime de bandă**. În practică, tăierea nu este verticală (și deci frecvența de tăiere nu este exactă), astfel încât deseori lărgimea de bandă este aproximată ca intervalul dintre 0 și frecvența de trecere pentru jumătate din puterea maximă.

Lărgimea de bandă este o proprietate fizică a mediului de transmisie și de obicei depinde de construcția, grosimea și lungimea mediului. În unele cazuri, în circuit este introdus un filtru pentru a limita lărgimea de bandă disponibilă pentru fiecare client. De exemplu, un fir de telefon poate avea lărgimea de bandă de 1 MHz pentru distanțe scurte, dar companiile telefonice adaugă un filtru ce limitează fiecare client la aproximativ 3100 Hz. Această variantă este adecvată pentru vorbire inteligibilă și îmbunătățește eficiența sistemului prin limitarea utilizării de resurse de către clienți.

Să vedem cum va arăta semnalul transmis dacă banda de frecvență folosită ar fi atât de îngustă, încât numai frecvențele foarte joase pot fi transmise [funcția ar fi aproximată doar cu primii câțiva termeni ai ecuației (2-1)]. Fig. 2-1(b) reprezintă semnalul rezultat dintr-un canal care permite numai primei armonici (*f*, fundamentală) să fie transmisă. Similar, fig. 2-1(c)-(e) prezintă spectrele și funcțiile reconstruite pentru canale cu lărgime de bandă mai mare.

Fiind dată o rată de transmisie a biților de *b* biți/secundă, timpul necesar pentru a transmite 8 biți (de exemplu) este de 8/*b* secunde, frecvența primei armonice fiind *b*/8 Hz. O linie telefonică obișnuită, deseori numită **linie în bandă vocală** (**voice-grade line**), este limitată artificial de o frecvență de tăiere puțin peste 3000 Hz. Această restricție impune ca numărul celei mai mari armonice care poate fi transmisă este aproximativ de 3000/(*b*/8), adică 24000/*b* (frecvența de prag nu este foarte exactă).

Bps	T (msec)	Prima armonică (Hz)	nr. armonice transmise
300	26.27	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Fig. 2-2. Relația între viteza de transfer a datelor și armonice.

În fig. 2-2 sunt prezentate valorile pentru anumite viteze de transfer de date. Pornind de la aceste valori, este clar că încercarea de a transmite date la o viteză de 9600 bps folosind o linie telefonică obișnuită va transforma semnalul din fig. 2-1(a) în ceva asemănător cu fig. 2-1(c), fiind dificilă obținerea secvenței de biți originale. Este evident că la viteze de transfer mai mari decât 38.4 Kbps nu

există nici o speranță de a recupera semnalele binare, chiar dacă mediul de transmisie ar fi lipsit în totalitate de zgomote. Cu alte cuvinte, limitând lărgimea de bandă se limitează și viteza de transfer chiar și pentru canalele perfecte. Oricum, există tehnici de codificare sofisticate, care folosesc mai multe niveluri de tensiune și care pot atinge rate de transfer mai mari. Vom discuta aceste tehnici mai târziu în acest capitol.

2.1.3 Viteza maximă de transfer de date a unui canal

Încă din 1924, un inginer AT&T, H. Nyquist a descoperit că și un canal perfect are o capacitate limitată de transmisie. El a dedus o ecuație care exprimă viteza maximă de transfer de date pentru un canal fără zgomote, cu lărgime de bandă finită. În 1948, Claude Shannon a continuat cercetările lui Nyquist exprimând această limită pentru un canal supus zgomotului aleatoriu (termodinamic) (Shannon 1948). Noi nu vom face aici decât o scurtă prezentare a acestor rezultate, acum devenite clasice.

Nyquist a demonstrat că dacă un semnal arbitrar este transmis printr-un filtru de frecvențe joase cu lărgime de bandă H, semnalul filtrat poate fi complet reconstruit prin efectuarea a numai $2H$ eșantioane pe secundă. Eșantionarea semnalului la o viteză mai mare decât $2H$/secundă este inutilă, deoarece componentele cu o frecvență mai înaltă pe care aceste eșantioane le-ar putea obține au fost deja filtrate. Dacă semnalul are V niveluri discrete, teorema lui Nyquist afirmă:

$$\text{viteza maximă de transfer de date} = 2H \log_2 V \text{ biți / sec}$$

De exemplu, un canal de 3kHz, fără zgomote, nu poate transmite semnale binare (pe două niveluri) la o viteză mai mare de 6000 bps.

Până acum am studiat doar cazul canalelor fără zgomote. Dacă sunt prezente zgomote aleatoare, situația se deteriorează rapid. Iar un zgomot aleator (termic) datorat mișcării moleculelor în sistem va fi prezent întotdeauna. Dimensiunea zgomotului termic prezent se măsoară prin raportul dintre puterea semnalului și puterea zgomotului, fiind numită **raportul semnal-zgomot**. Dacă notăm puterea semnalului cu S și puterea zgomotului cu N, atunci raportul semnal-zgomot este S/N. De obicei, acest raport nu este specificat; în schimb, este dată expresia $10 \log_{10} S/N$. Aceste unități sunt numite **decibeli** (dB). Un raport S/N egal cu 10 este de 10 dB, un raport egal cu 100 este de 20 dB, un raport egal cu 1000 este de 30 dB și așa mai departe. De multe ori fabricanții de amplificatoare stereo caracterizează banda de frecvență (domeniul de frecvență) în care produsul lor este liniar furnizând frecvențele la care semnalul se atenuează cu 3 dB la fiecare capăt. Acestea sunt punctele în care factorul de amplificare este aproximativ înjumătățită (deoarece $\log_{10} 3 \approx 0.5$).

Rezultatul cel mai important obținut de Shannon este expresia pentru viteza maximă de transfer de date a unui canal cu zgomote, având lărgimea de bandă de H Hz și a cărui raport semnal-zgomot S/N este dat de:

$$\text{numărul maxim de biți/sec} = H \log_2 (1 + S/N)$$

De exemplu, un canal cu o bandă de frecvență de 3000 Hz și zgomot termic de 30 dB (parametri tipici părții analogice a sistemului telefonic) nu va putea transmite mult mai mult de 30.000 bps, indiferent de cât de multe sau de puține niveluri are semnalul sau cât de multe sau puține eșantioane sunt luate. Rezultatele lui Shannon au fost obținute folosind atât argumente teoretice cât și argumente informaționale și se aplică oricărui canal supus zgomotelor termice. Contraexemplele ar tre-

bui plasate în aceeași categorie cu mașinile perpetuum mobile. Ar trebui remarcat și că această viteză nu este decât o limitare superioară pe care sistemele reale o ating rareori.

2.2 MEDII DE TRANSMISIE GHIDATĂ

Scopul nivelului fizic este de a transporta o secvență de biți de la o mașină la alta. Pentru transmisia efectivă pot fi utilizate diverse medii fizice. Fiecare dintre ele este definit de lărgime proprie de bandă, întârziere, cost, dar și de ușurința de instalare și întreținere. Aceste medii pot fi împărțite în două grupe mari: medii ghidate, cum sunt cablul de cupru și fibrele optice și medii neghidate, cum sunt undele radio și laserul. Vom arunca o privire asupra acestora în următoarele secțiuni.

2.2.1 Medii magnetice

Una din cele mai obișnuite metode de a transporta date de la un calculator la altul este să se scrie datele pe o bandă magnetică sau pe un suport reutilizabil (de exemplu, DVD-uri pentru înregistrare), să se transporte fizic banda sau discul la mașina de destinație, după care să se citească din nou datele. Cu toate că această metodă nu este la fel de sofisticată precum folosirea unui satelit de comunicație geosincron, ea este de multe ori mai eficientă din punct de vedere al costului, mai ales pentru aplicațiile în care lărgimea de bandă sau costul pe bit transportat sunt factori cheie.

Un calcul simplu va confirma acest punct de vedere. O bandă Ultrium standard industrial poate înmagazina 200 gigaocteți. Într-o cutie cu dimensiunile 60 x 60 x 60 cm pot să încapă cam 1000 de astfel de benzi, ceea ce înseamnă o capacitate totală de 200 de terraocteți sau 1600 terrabiți (1.6 petabiți). O cutie cu benzi poate fi distribuită oriunde în Statele Unite în 24 de ore de către Federal Express sau de alte companii. Banda de frecvență efectivă a acestei transmisii este de 1600 terrabiți / 84600 secunde, adică 19Gbps. Dacă destinația ar fi la distanță de numai o oră cu mașina, lărgimea de bandă s-ar mări la peste 400Gbps. Nici o rețea de calculatoare nu poate să se apropie de o asemenea viteză.

Pentru o bancă în care datele sunt de ordinul gigaoctecților și trebuie salvate zilnic pe o altă mașină (pentru ca banca să poată funcționa în continuare chiar și în urma unor inundații puternice sau unui cutremur), probabil că nici o altă tehnologie de transmisie nu e comparabilă cu performanța atinsă de banda magnetică. Desigur, rețelele devin din ce în ce mai rapide, dar și capacitățile benzilor magnetice cresc.

Dacă ne uităm la cost, vom obține aceeași situație. Atunci, dacă sunt cumpărate en-gros, benzile Ultrium ajung să coste în jur de 40 de dolari pe bucată. O bandă poate fi refolosită de cel puțin 10 ori, astfel încât costul benzii este aproape de 4000 de dolari/cutie/utilizare. Dacă adăugăm încă 1000 de dolari pentru transport (probabil mult mai ieftin), vom avea un cost de 5000 de dolari pentru a transporta 200 de terraocteți. De aici rezultă că un gigaoctet poate fi transportat la un preț mai mic de 3 cenți. Nici o rețea nu poate concura cu un astfel de preț. Morala poveștii:

Niciodată nu subestima lărgimea de bandă a unui camion încărcat cu benzi magnetice care gonește la vale pe autostradă.

2.2.2 Cablul torsadat

Deși caracteristicile de lărgime de bandă ale mediilor magnetice sunt excelente, performanțele legate de întârzieri sunt slabe. Timpul de transmisie nu se măsoară în milisecunde, ci în minute sau ore. Pentru multe aplicații este nevoie de o conexiune on-line. Unul dintre cele mai vechi medii de transmisie, rămas cel mai utilizat mediu, este **cablul torsadat**. O pereche torsadată este formată din două fire de cupru izolate, fiecare având o grosime tipică de 1 mm. Firele sunt împletite într-o formă elicoidală, ca o moleculă de ADN. Împletirea se face pentru că două fire paralele constituie o bună antenă. Dacă firele sunt împletite, undele din diferite împletiri se anulează, astfel încât radiația firului este scăzută eficient.

Cea mai cunoscută aplicație a cablului torsadat este sistemul telefonic. Aproape toate telefoanele sunt conectate la centrala telefonică printr-un cablu torsadat. Cablurile torsadate se pot întinde pe mai mulți kilometri fără amplificare, dar pentru distanțe mai mari, sunt necesare repetoare. Atunci când mai multe cabluri torsadate sunt grupate în paralel – cum sunt de exemplu toate firele de la un bloc de locuințe legate la centrala telefonică – ele sunt legate împreună și încapsulate într-un material protector. Dacă perechile de fire nu ar fi fost împletite, cablurile grupate astfel împreună ar fi interferat. În anumite părți ale lumii, unde liniile telefonice sunt montate pe stâlpi, sunt des întâlnite cablurile cu diametrul de câțiva centimetri.

Cablurile torsadate pot fi folosite atât pentru transmisia semnalelor analogice cât și pentru transmisia de semnale digitale. Lărgimea de bandă depinde de grosimea firului și de distanța parcursă, dar, în multe cazuri, se poate atinge o viteză de mai mulți megabiți pe secundă pe distanțe de ordinul a câțiva kilometri. Datorită performanței satisfăcătoare și a costului scăzut, cablurile torsadate sunt foarte larg folosite în prezent și probabil că vor rămâne larg folosite și în următorii ani.

Există numeroase tipuri de cabluri torsadat, două dintre acestea fiind importante pentru rețelele de calculatoare. Perechile torsadate din **Categoria 3** sunt formate din două fire izolate răsucite unul în jurul celuilalt cu pas mare. De obicei, patru astfel de perechi sunt grupate într-un material plastic, pentru a le proteja și pentru a le ține împreună. Până în 1988, cele mai multe clădiri cu birouri aveau un cablu de categoria 3, care pornea din panoul central de la fiecare etaj către fiecare birou. Această schemă permitea ca maxim patru telefoane obișnuite, sau maxim două telefoane cu mai multe linii, toate aflate în același birou, să poată fi cuplate la centrala telefonică prin panoul central.

Începând din 1988, au fost introduse cablurile de **Categoria 5**, mai performante. Ele sunt similare celor din categoria 3, dar au mai multe răsuciri pe centimetru (pas de răsucire mai mic), rezultând o interferență (diafonie) scăzută și o mai bună calitate a semnalului pe distanțe mari, ceea ce le face mai adecvate comunicațiilor la viteze mari între calculatoare. Categoriile mai noi sunt 6 și 7, care sunt capabile să trateze semnale cu banda de frecvență de 250 MHz și, respectiv, 600 MHz (față de numai 16MHz sau 100MHz pentru categoriile 3 și, respectiv, 5) .

Pentru a le deosebi de cablurile torsadate voluminoase, ecranate și scumpe, pe care IBM le-a introdus la începutul anilor ‘80, dar care nu au devenit populare în afara instalațiilor IBM, aceste tipuri de cabluri sunt cunoscute sub numele de cabluri UTP (Unshielded Twisted Pair, rom: cablu torsadat neecranat). Torsadarea firelor este ilustrată în fig. 2-3.

(a) (b)

Fig. 2-3. (a) Cablu UTP cat. 3. (b) Cablu UTP cat. 5.

2.2.3 Cablu Coaxial

Un alt mediu uzual de transmisie este cablul coaxial (cunosc ut printre utilizatorii săi sub numele de „coax” și este pronunțat “co-ax”). El are o ecranare mai bună decât cablurile torsadate, putând acoperi distanțe mai mari la rate de transfer mai mari. Există două tipuri de cabluri coaxiale folosite pe scară largă. Primul, cablul de 50 de ohmi, este folosit frecvent când se dorește transmisie digitală de la început. Al doilea tip, cablul de 75 de ohmi, este frecvent folosit în transmisia analogică și televiziunea prin cablu, dar devine tot mai important o data cu apariția Internetului prin cablu. Această clasificare are la bază un criteriu stabilit mai mult pe considerente istorice decât pe considerente tehnice (de exemplu, primele antene dipol aveau o impedanță de 300 de ohmi și existau transformatoare de impedanță 4 : 1, care erau ușor de folosit).

Un cablu coaxial este format dintr-o sârmă de cupru rigidă, protejată de un material izolator. Acest material este încapsulat într-un conductor circular, de obicei sub forma unei plase strâns întrețesute. Conductorul exterior este acoperit cu un înveliș de plastic protector. În fig. 2-4 este prezentată o vedere în secțiune a cablului coaxial.

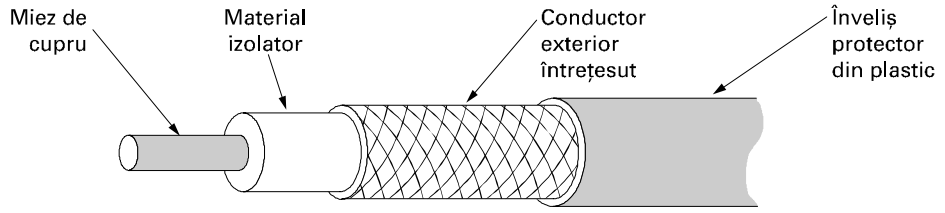


Fig. 2-4. Un cablu coaxial.

Structura și ecranarea cablului coaxial asigură o bună împletire a necesităților semnificative de lărgime de bandă și totodată de imunitate excelentă la zgomot. Lărgimea de bandă poate depinde de calitatea cablului, de lungime, și de raportul semnal-zgomot al semnalului de date. Cablurile moderne au o bandă de frecvență de aproape 1 GHz. Cablurile coaxiale erau folosite pe scară largă în sistemul telefonic pentru linii întinse pe distanțe mari, dar au fost în mare parte înlocuite cu fibre optice. Oricum, cablul coaxial este utilizat în continuare în televiziunea prin cablu și în unele rețele locale.

2.2.4 Fibre optice

Mulți dintre cei implicați în industria calculatoarelor sunt foarte mândri de viteza de evoluție a tehnologiei calculatoarelor. Originalul (1981) IBM PC rula la o frecvență de ceas de 4,77 MHz. Douăzeci de ani mai târziu, calculatoarele personale pot rula la 2 GHz, ceea ce reprezintă o creștere a frecvenței de 20 de ori pentru fiecare deceniu. Nu e rău deloc.

În aceeași perioadă, comunicațiile de date pe arii întinse au evoluat de la o viteză de 56 Kbps (ARPANET) până la 1 Gbps (comunicațiile optice moderne), o creștere de mai bine de 125 de ori pentru fiecare deceniu. În aceeași perioadă, frecvența erorilor a scăzut de la 10^{-5} per bit până aproape de zero.

Mai mult, procesoarele se apropie de limitele lor fizice, date de viteza luminii și de problemele de disipare a căldurii. Din contră, folosind tehnologiile *actuale* de fibre optice, lărgimea de bandă care poate fi atinsă este cu siguranță mai mare decât 50,000 Gbps (50 Tbps) și sunt încă mulți cei

care caută materiale și tehnologii mai performante. Limitarea practică actuală la aproximativ 10 Gbps este o consecință a imposibilității de a converti mai rapid semnalele electrice în semnale optice, deși, în laborator, 100 Gbps a fost atinsă într-o singură fibră.

În cursa dintre calculatoare și comunicații, acestea din urmă au învins. Implicațiile complete ale lărgimii de bandă infinite (deși nu la un cost nul) nu au fost încă abordate de o generație de oameni de știință și ingineri învățați să gândească în termenii limitărilor calculate de Nyquist și Shannon, limitări stricte impuse de firele de cupru. Noua paradigmă convențională spune că mașinile de calcul sunt extrem de lente, astfel că rețelele trebuie să evite cu orice preț calculele, indiferent de lărgimea de bandă risipită. În această secțiune vom studia fibrele optice pentru a ne familiariza cu această tehnologie de transmisie.

Un sistem de transmisie optică este format din trei componente: sursa de lumină, mediul de transmisie și detectorul. Prin convenție, un impuls de lumină înseamnă un bit cu valoarea 1, iar absența luminii indică un bit cu valoarea 0. Mediul de transmisie este o fibră foarte subțire de sticlă. Atunci când interceptează un impuls luminos, detectorul generează un impuls electric. Prin atașarea unei surse de lumină la un capăt al fibrei optice și a unui detector la celălalt, obținem un sistem unidirecțional de transmisie a datelor care primește un semnal electric, îl convertește și îl transmite ca impulsuri luminoase și apoi reconvertește ieșirea în semnale electrice la recepție.

Acest sistem de transmisie ar fi pierdut din semnalele luminoase și ar fi fost lipsit de importanță în practică, dacă nu s-ar fi folosit un principiu interesant al fizicii: când o rază luminoasă trece de la un mediu la altul, de exemplu de la siliciu la aer, raza este refractată (frântă) la suprafața de separație siliciu / aer ca în fig. 2-5. Se observă o rază de lumină incidentă pe suprafața de separație la un unghi α_1 care se refractă la un unghi β_1. Mărimea refracției depinde de proprietățile celor două medii (în particular, de indicii lor de refracție). Pentru unghiuri de incidență mai mari decât o anumită valoare critică, lumina este refractată înapoi în siliciu fără nici o pierdere. Astfel o rază de lumină, la un unghi egal sau mai mare decât unghiul critic, este încapsulată în interiorul fibrei, ca în fig. 2-5(b) și se poate propaga pe mulți kilometri, aparent fără pierderi.

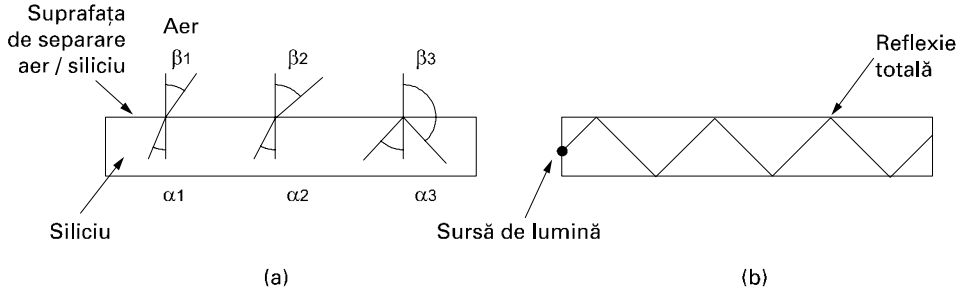


Fig. 2-5. (a) Trei exemple de raze de lumină în interiorul unei fibre de siliciu care cad pe suprafața de separație aer/siliciu la unghiuri diferite. (b) Încapsularea luminii prin reflexie totală.

În fig. 2-5(b) se poate observa o singura rază încapsulată, dar se pot transmite mai multe raze cu unghiuri de incidență diferite, datorită faptului că orice rază de lumină cu unghi de incidență la suprafața de separație mai mare decât unghiul critic va fi reflectată total. Se spune că fiecare rază are un **mod** diferit, iar fibra care are această proprietate se numește **fibră multi-mod**.

Oricum, dacă diametrul fibrei este redus la câteva lungimi de undă ale luminii, fibra acționează ca un ghid de undă și lumina se va propaga în linie dreaptă, fără reflexii, rezultând o **fibră monomod**. Aceste fibre sunt mai scumpe, dar sunt des folosite pentru distanțe mai mari. Fibrele mono-

mod curente pot transmite date la 50 Gbps pe distanțe de 100 Km fără amplificare. În condiții de laborator și pentru distanțe mai mici s-au obținut rate de transfer chiar și mai mari.

Transmisia luminii prin fibre

Fibrele optice sunt fabricate din sticlă, iar sticla este fabricată la rândul ei din nisip, un material brut necostisitor, care se găsește în cantități nelimitate. Producerea sticlei era cunoscută de egiptenii din Antichitate, dar pentru ei sticla trebuia să nu fie mai groasă de 1 mm pentru ca lumina să poată să treacă prin ea. Sticla suficient de transparentă pentru a putea fi folosită ca fereastră a apărut abia în timpul Renașterii. Sticla folosită pentru fibrele optice moderne este atât de transparentă încât, dacă oceanele ar fi fost pline cu astfel de sticlă în loc de apă, fundul oceanului s-ar vedea de la suprafață tot atât de clar precum se vede pământul din avion într-o zi senină.

Atenuarea luminii prin sticlă depinde de lungimea de undă a luminii (și de alte câteva proprietăți fizice ale sticlei). Pentru tipul de sticlă folosit la fibre optice, atenuarea este prezentată în fig. 2-6, măsurată în decibeli pe kilometru liniar de fibră. Atenuarea în decibeli este dată de formula:

$$Atenuarea\_în\_decibeli = 10\log_{10}\frac{puterea\_transmisă}{puterea\_receptionată}$$

De exemplu, pentru un factor de pierdere egal cu 2 rezultă o atenuare de $10 \log_{10}2 = 3$ dB. Fig. prezintă valorile atenuării pentru lungimi de undă apropiate spectrului razelor infraroșii, care sunt folosite în practică. Lumina vizibilă are lungimi de undă puțin mai mici, de la 0.4 la 0.7 microni (1 micron este 10^{-6} metri).

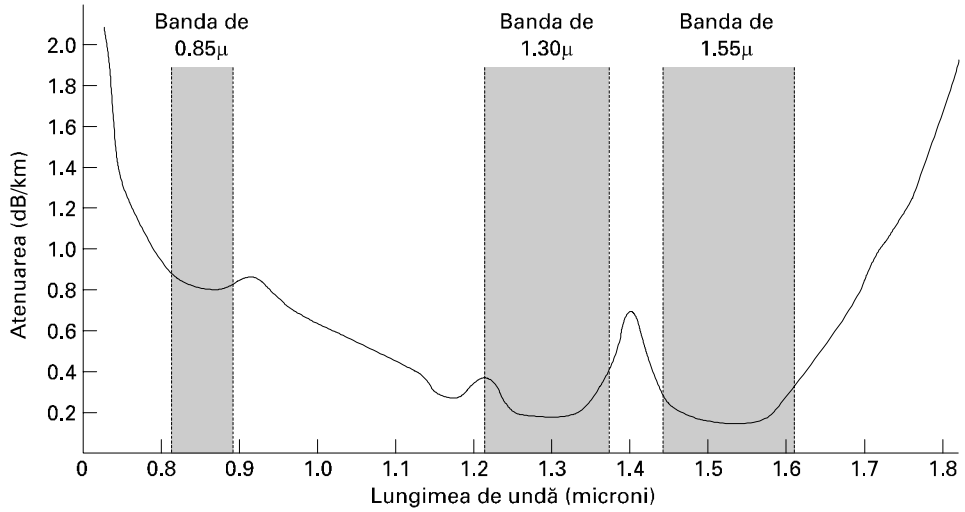


Fig. 2-6. Atenuarea luminii prin fibră în spectrul infraroșu.

Trei benzi din acest spectru sunt folosite în comunicații. Ele sunt centrate respectiv la 0.85, 1.3 și 1.55 microni. Ultimele două au proprietăți bune de atenuare (mai puțin de 5% pierderi pe kilometru). Banda de 0.85 microni are o atenuare mai mare, dar o proprietate care o avantajează este că, la această lungime de undă, laserul și echipamentul electronic pot fi făcute din același material (arseniură de galiu). Toate cele trei benzi au o lărgime de bandă între 25.000 și 30.000 GHz.

Impulsurile de lumină transmise prin fibră își extind lungimea în timpul propagării. Această extindere se numește **dispersie cromatică**, și mărimea ei este dependentă de lungimea de undă. Un mod de a preveni suprapunerea acestor impulsuri extinse este de a mări distanța dintre ele, dar aceasta se poate face doar prin reducerea ratei semnalului. Din fericire, s-a descoperit că, dând acestor impulsuri o formă specială, legată de reciproca cosinusului hiperbolic, se anulează toate efectele de dispersie, și este astfel posibil să se trimită impulsuri pe mii de kilometri, fără distorsiuni semnificative ale formei. Aceste impulsuri se numesc **solitonuri**. Cercetările pentru implementarea practică a acestei soluții de laborator sunt în plină desfășurare.

Cablurile din fibră optică

Cablurile din fibră optică sunt similare celor coaxiale, cu singura deosebire că nu prezintă acel material conductor exterior sub forma unei plase. Fig. 2-7(a) prezintă o secțiune a unei singure fibre. În centru se află miezul de sticlă prin care se propagă lumina. În fibrele multi-mod, miezul are un diametru de 50 microni, aproximativ grosimea părului uman. În fibrele mono-mod miezul este de 8 până la 10 microni.

Miezul este îmbrăcat în sticlă cu un indice de refracție mai mic decât miezul, pentru a păstra lumina în miez. Totul este protejat cu o învelitoare subțire din plastic. De obicei, mai multe fibre sunt grupate împreună, protejate de o teacă protectoare. Fig. 2-7(b) prezintă un astfel de cablu cu trei fibre.

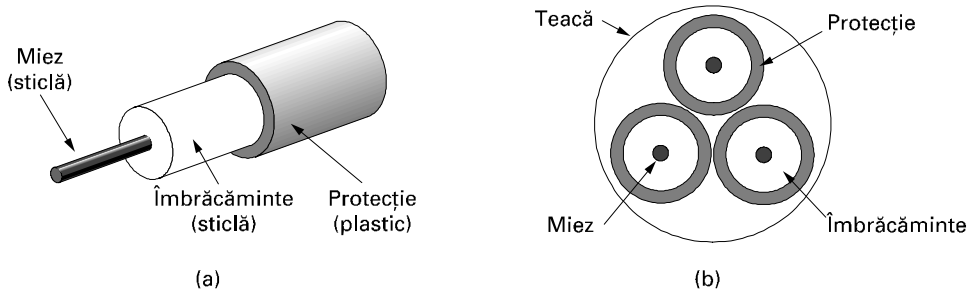


Fig. 2-7 (a) Vedere perspectivă a unei singure fibre.
(b) Vedere în secțiune a unei teci cu trei fibre.

Fibrele terestre sunt îngropate în pământ până la adâncimi de un metru, fiind ocazional deteriorate de buldozere sau de cârtițe. Lângă țărm, fibrele transoceanice sunt îngropate în șanțuri cu ajutorul unui fel de plug de mare. În apele adânci, ele stau pe fundul apei, unde pot fi agățate de traulere de pescuit sau pot fi atacate de calmari. Fibrele pot fi conectate în trei moduri. Primul mod constă în atașarea la capătul fibrei a unor conectori care se pot lega la un soclu pentru fibră. Conectorii pierd între 10% și 20% din lumină, dar avantajul acestor sisteme este că sunt ușor de reconfigurat.

Al doilea mod constă în îmbinarea mecanică. Îmbinările mecanice se obțin prin atașarea celor două capete unul lângă altul, într-un înveliș special, și fixarea lor cu ajutorul unor clame. Alinierea se poate face prin trimitere de semnale prin joncțiune și realizarea de mici ajustări pentru a maximiza semnalul. Unui specialist îi trebuie în jur de 5 minute să facă o îmbinare mecanică, aceasta având ca rezultat o pierdere de lumină de 10%.

A treia posibilitate este de a îmbina (topi) cele două bucăți de fibră, pentru a forma o conexiune solidă. O îmbinare prin sudură este aproape la fel de bună ca și folosirea unui singur fir, dar chiar și aici, apare o mică atenuare.

Pentru toate cele trei tipuri de îmbinare poate să apară fenomenul de reflexie la punctul de îmbinare, iar energia reflectată poate interfera cu semnalul.

Criteriu	LED	Laser cu semiconductor
Viteza de transfer a datelor	Joasă	Mare
Tip de fibră	Multi-mod	Multi-mod sau uni-mod
Distanță	Scurtă	Lungă
Durata de viață	Viață lungă	Viață scurtă
Sensibilitate la temperatură	Minoră	Substanțială
Cost	Cost redus	Scump

Fig. 2-8. O comparație între laserele semiconductoare și LED-uri ca surse de lumină.

Pentru transmiterea semnalului se pot folosi două tipuri de surse de lumină: LED-uri (Light Emitting Diode – diodă cu emitere de lumină) și laserul cu semiconductor. Ele au proprietăți diferite, după cum arată fig. 2-8. Ele se pot ajusta în lungime de undă prin introducerea interferometrelor Fabry-Perot sau Mach-Zender între sursă și fibra optică. Interferometrele Fabry-Perot sunt simple cavități rezonante, formate din două oglinzi paralele. Lumina cade perpendicular pe oglinzi. Lungimea acestei cavități selectează acele lungimi de undă care încap în interior de un număr întreg de ori. Interferometrele Mach-Zender separă lumina în două fascicole. Cele două fascicole se propagă pe distanțe ușor diferite. Ele sunt apoi recombinate și se află în fază doar pentru anumite lungimi de undă.

Capătul fibrei optice care recepționează semnalul constă dintr-o fotodiodă, care declanșează un impuls electric când primește o rază de lumină. Timpul de răspuns tipic al unei diode este de 1ns, ceea ce limitează viteza de transfer de date la aproximativ 1Gbps. Pentru a putea fi detectat, un impuls luminos trebuie să aibă suficientă energie ca să evite problema zgomotului termic. Viteza de apariție a erorilor se poate controla prin asocierea unei puteri suficient de mari a semnalului.

Rețelele din fibre optice

Fibrele optice pot fi folosite atât pentru LAN-uri cât și pentru transmisia pe distanțe foarte lungi, deși conectarea într-o rețea bazată pe acest mediu este mult mai complexă decât conectarea la Ethernet. O soluție pentru a evita această problemă este prezentarea unei rețele în inel ca fiind o colecție de legături punct la punct, așa ca în fig. 2-9. Interfața fiecărui calculator lasă să treacă impulsul de lumină către următoarea legătură și totodată are rolul unei joncțiuni în T pentru a face posibilă transmiterea și recepția mesajelor.

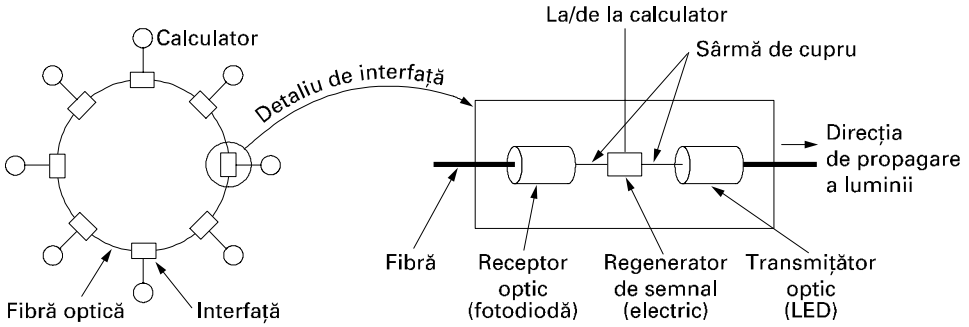


Fig. 2-9. Un inel din fibră optică cu repetoare active.

Se folosesc două tipuri de interfețe. O interfață activă constă din doi conectori sudați pe fibra centrală. Unul din ei are la un capăt un LED sau o diodă cu laser (pentru transmisie) și celălalt are la capăt o fotodiodă (pentru recepție). Conectorul este complet pasiv și este viabil, deoarece un LED sau o fotodiodă defectă nu întrerupe inelul, ci doar scoate un calculator din circuit.

Un alt model de interfață, prezentat în fig. 2-9, este **repetorul activ**. Lumina recepționată este convertită într-un semnal electric, regenerat la putere maximă, dacă este atenuat și retransmis ca semnal luminos. Interfața cu calculatorul este un fir de cupru obișnuit care se leagă la regeneratorul de semnal. În prezent, sunt folosite și repetoare integral optice. Aceste echipamente nu necesită conversii de tipul optic-electric-optic, ceea ce înseamnă că pot opera la lărgimi de bandă foarte mari.

În cazul în care repetorul activ se deteriorează, inelul este întrerupt și rețeaua nu mai funcționează. Pe de altă parte, deoarece semnalul este regenerat de fiecare interfață, legăturile între două calculatoare adiacente pot avea lungimi de kilometri, practic fără nici o limitare asupra dimensiunii totale a inelului. Interfețele pasive diminuează lumina la fiecare joncțiune, având ca efect restricții drastice în ceea ce privește numărul de calculatoare ce pot fi conectate și lungimea totală a inelului.

O topologie în inel nu este singura modalitate de a construi un LAN folosind fibre optice. Este posibilă și o arhitectură de tip **stea pasivă**, ca aceea prezentată în fig. 2-10. În această schemă, fiecare interfață prezintă o fibră care face conexiunea între transmițător și un cilindru de siliciu, cu toate aceste fibre sudate la un capăt al cilindrului. Similar, fibrele sudate la celălalt capăt al cilindrului se conectează la fiecare receptor. Ori de câte ori o interfață transmite un semnal, el este difuzat în interiorul stelei pasive pentru a ilumina toți receptorii, realizându-se astfel difuzarea. Steaua pasivă combină toate semnalele de la intrare și transmite semnalul combinat pe toate liniile. Deoarece energia de la intrare este împărțită între toate liniile de la ieșire, numărul de noduri în rețea este limitat de senzitivitatea fotodiodelor.

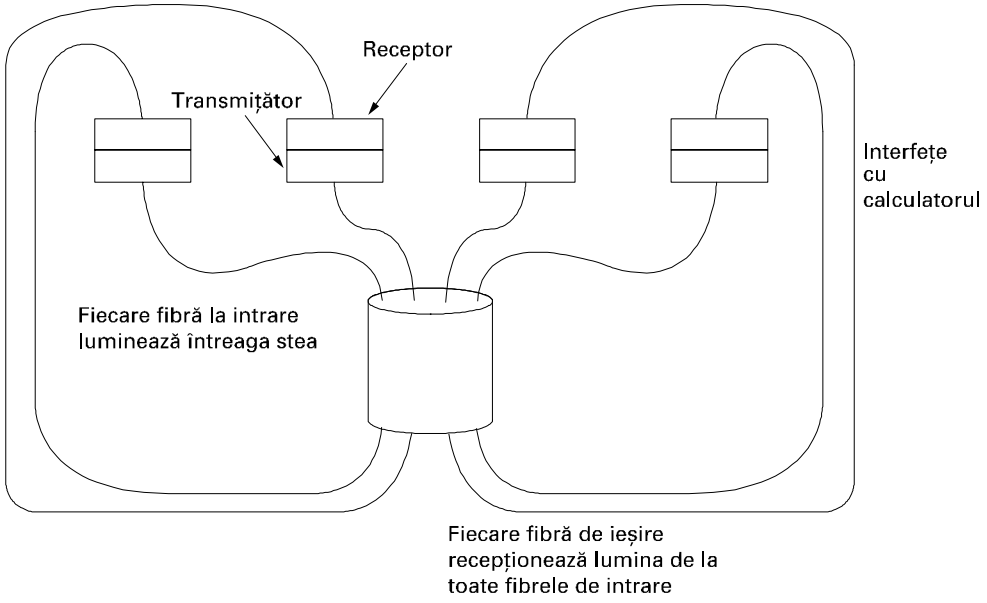


Fig. 2-10. Conectarea unei stele pasive în rețelele de fibră optică.

Comparație între fibrele optice și firul de cupru

O comparație între fibra optică și firele de cupru este instructivă. Fibra are multe avantaje. Mai întâi, lărgimea de bandă pe care o suportă este mai mare decât a firelor de cupru. Numai acest lucru și ar fi suficient pentru a fi utilizată în rețelele performante. Datorită atenuării scăzute, repetoarele sunt necesare la fiecare 30 km pe liniile lungi, în comparație cu 5 Km pentru cupru. Fibra are avantajul că nu este afectată de șocurile electrice, de interferența câmpului electromagnetic sau de căderile de tensiune. De asemenea, nu este afectată nici de substanțele chimice corozive din aer, fiind ideală pentru mediile aspre din fabrici.

Destul de surprinzător, companiile de telefoane preferă fibra dintr-un alt motiv: este subțire și foarte ușoară. Multe dintre canalele cu cabluri sunt pline până la refuz și prin înlocuirea cuprului cu fibră se obține ceva spațiu, iar cuprul are o valoare excelentă pe piață, deoarece fabricile îl consideră un minereu de mare importanță. De asemenea, fibra este mai ușoară decât cuprul. O mie de cabluri torsadate cu o lungime de 100 Km lungime cântăresc 8000 Kg. Două fibre au o capacitate mai mare și cântăresc doar 100 Kg, acest lucru reducând drastic necesitatea unor echipamente mecanice scumpe care trebuie întreținute. Pe traseele noi, fibra câștigă detașat în fața cuprului datorită costului de instalare foarte scăzut.

În sfârșit, fibrele nu disipă lumina și de aceea sunt foarte dificil de interceptat. Aceste proprietăți le oferă o excelentă securitate împotriva unor potențiale tentative de interceptare.

Pe de altă parte, fibra este o tehnologie mai puțin familiară și necesită o pregătire pe care nu toți inginerii o au, iar fibrele pot fi stricate ușor dacă sunt îndoite prea mult. Deoarece transmisia optică este prin natura ei unidirecțională, comunicațiile bidirecționale necesită fie două fibre, fie două benzi de frecvență diferite pe aceeași fibră. În sfârșit, interfețele pentru fibră costă mai mult decât interfețele electrice. Nu mai puțin adevărat este faptul că toate comunicațiile de date pe lungimi mai mari de câțiva metri se vor face în viitor cu fibre. Pentru o discuție asupra tuturor aspectelor fibrelor optice și asupra rețelelor construite cu ele, vedeți (Hecht, 2001).

2.3 COMUNICAȚIILE FĂRĂ FIR

Epoca noastră a generat dependența de informație: oameni care au nevoie să fie în permanența conectați la informații. Pentru acești utilizatori mobili, cablul torsadat, cablul coaxial și fibrele optice nu sunt de nici un folos. Ei au nevoie de date pentru calculatoarele lor portabile, fără a fi legați de infrastructura comunicațiilor terestre. Pentru acești utilizatori comunicațiile fără fir reprezintă soluția optimă. În secțiunile ce urmează, vom discuta la modul general asupra comunicațiilor fără fir, deoarece acestea au multe alte aplicații importante în afara serviciilor de conectare oferite utilizatorilor care doresc să navigheze pe WEB de pe plajă.

Sunt voci care susțin că viitorul rezervă numai două tipuri de comunicații: prin fibre optice și fără fir. Toate calculatoarele, faxurile, telefoanele fixe (nemobile) vor folosi fibre, iar cele mobile vor folosi comunicația fără fir.

Comunicațiile fără fir sunt avantajoase chiar și pentru echipamentele fixe, în anumite împrejurări. De exemplu, în cazul în care conectarea unei clădiri cu ajutorul fibrei este dificilă datorită terenului (munți, jungle, mlaștini etc.), comunicația fără fir poate fi mai bună. Este de remarcat faptul că

sistemele moderne de comunicație digitală fără fir au apărut în Insulele Hawaii, unde utilizatorii erau despărțiți de mari întinderi de apă din oceanul Pacific, sistemul telefonic fiind inadecvat.

2.3.1 Spectrul electromagnetic

Atunci când electronii se află în mișcare, ei creează unde electromagnetice care se pot propaga prin spațiu (chiar și în vid). Aceste unde au fost prezise de fizicianul britanic James Clerk Maxwell în 1865 și au fost observate pentru prima dată de fizicianul german Heinrich Hertz în 1887. Numărul de oscilații ale unei unde într-o secundă poartă numele de **frecvență**, *f*, și este măsurată în **Hz** (în onoarea lui Heinrich Hertz). Distanța dintre două maxime (sau minime) consecutive este numită **lungime de undă.** Notația universală a lungimii de undă este λ (lambda).

Când o antenă dimensionată corespunzător este atașată unui circuit, undele electromagnetice pot fi difuzate eficient și interceptate de un receptor, aflat la o anumită distanță. Acest principiu stă la baza tuturor comunicațiilor fără fir.

În vid, toate undele electromagnetice se transmit cu aceeași viteză, indiferent de frecvență, Această viteză, numită de obicei **viteza luminii**, *c*, este de aproximativ de 3×10^8 m/s, sau aproape 1 picior (30 cm) pe nanosecundă. (Ar fi o ideea redefinirea *piciorului (*eng: *foot*) ca fiind distanța pe care o parcurge lumina în vid într-o nanosecunda, mai degrabă decât definirea pe baza mărimii pantofului unui rege oarecare mort demult). În cupru sau în fibră, viteza scade la aproape 2/3 din această valoare și devine ușor dependentă de frecvența undei. Viteza luminii este viteza maximă care poate fi atinsă – nici un obiect sau semnal nu se deplasează vreodată cu o viteza mai mare ca aceasta.

Relația fundamentală dintre *f*, λ și *c* (în vid) este

$$\lambda f = c \tag{2-2}$$

Deoarece *c* este o constantă, știind *f* putem afla λ, dar și invers. Ca o regulă clară, rețineți că atunci când λ este în metri și *f* este în MHz, λf=300. De exemplu, undele cu frecvența de 100 MHz au lungimea de undă de aproape 3 metri, cele cu frecvența de 1000 MHz au lungimea de undă de 0.3 metri, iar cele cu lungimea de undă de 0.1 metri au frecvența de 3000 MHz.

În fig. 2-11 este prezentat spectrul electromagnetic. Domeniile corespunzătoare undelor radio, microundelor, undelor infraroșii și luminii vizibile din spectru pot fi folosite pentru transmiterea informației prin modularea amplitudinii, frecvenței sau fazei undelor. Lumina ultravioletă, razele X și razele gama ar fi chiar mai performante datorită frecvențelor lor mai înalte, dar ele sunt greu de produs și de modulat, nu se propagă bine prin clădiri și sunt periculoase pentru ființele vii. Benzile listate în partea de jos a fig. 2-11 sunt numele oficiale ITU și se bazează pe lungimile de undă, LF acoperind intervalul de la 1 Km la 10 Km (aproximativ de la 30 KHz la 300 KHz). Termenii de LF, MF și HF se referă la frecvențele joase, medii și înalte, respectiv. Este evident că atunci când au fost date aceste nume, nimeni nu se aștepta ca frecvențe mai mari de 10 MHz să se folosească vreodată. Benzile mai înalte au fost numite mai târziu benzi de frecvență Foarte, Ultra, Super, Extrem și Extraordinar de înalte. Dincolo de aceste frecvențe nu mai există denumiri consacrate, dar am putea să folosim expresii de genul frecvențe Incredibil, Uimitor sau Miraculos de înalte.

Cantitatea de informație pe care o undă electromagnetică o poate transporta este legată de lărgimea ei de bandă. Folosind tehnologia curentă, este posibil să codificăm câțiva biți pe Hertz la frecvențe joase și deseori până la 8 biți pe Hertz la frecvențe înalte; în concluzie, un cablu torsadat cu lărgimea de bandă de 750MHz poate transporta date de ordinul a câțiva gigabiți/s. Din fig. 2-11 ar trebui să reiasă de acum foarte clar de ce profesioniștii din domeniul rețelelor apreciază atât de mult fibrele optice.

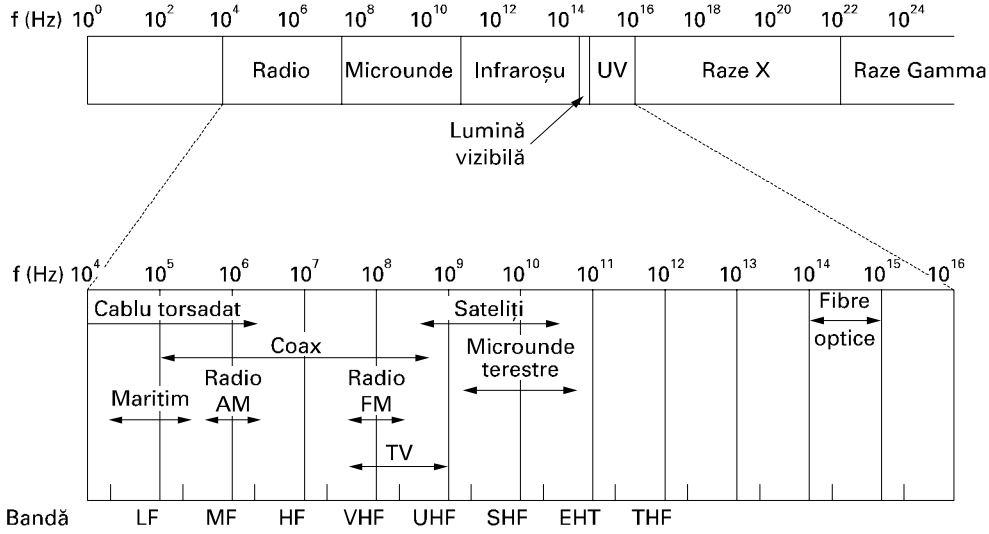


Fig. 2-11. Spectrul electromagnetic așa cum este folosit în comunicații.

Dacă rezolvăm Ec. (2-2) pentru f și o diferențiem în raport cu lungimea de undă, obținem

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2}$$

Dacă trecem la diferențe finite în loc de diferențiale și alegem doar valorile pozitive, obținem:

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2} \tag{2-3}$$

Astfel, fiind dată lărgimea unei benzi de lungimi de undă, $\Delta\lambda$, putem calcula banda de frecvență corespunzătoare, Δf și, în continuare, viteza de transfer de date pe care banda o poate produce. Cu cât banda este mai largă, cu atât crește viteza de transfer a datelor. De exemplu, să considerăm banda de 1.30 microni din fig. 2-6. Aici avem $\lambda = 1.3 \times 10^{-6}$ și $\Delta\lambda = 0.17 \times 10^{-6}$, cu Δf aproape 30 THz. La , să zicem, 8 biți/Hz, avem 240 Tbps.

Majoritatea transmisiilor folosesc o bandă îngustă de frecvență ($\Delta f / f << 1$) pentru a obține cea mai bună recepție (cât mai mulți Watts/Hz). Totuși, banda largă este folosită în două situații. În cazul utilizării metodei **salturilor de frecvență într-un spectru larg** (**frequency hopping spread spectrum**), transmițătorul sare de la o frecvență la alta de sute de ori pe secundă. Ea este foarte populară în comunicațiile armatei, deoarece face transmisia greu de detectat și aproape imposibil de bruiat. De asemenea, oferă un grad ridicat de imunitate la atenuarea multi-căi, deoarece semnalul direct ajunge întotdeauna primul la receptor. Semnalele reflectate urmăresc o cale mai lungă si ajung mai târziu. Până atunci receptorul a schimbat deja frecvența și nu mai primește semnale cu frecvența anterioară, eliminând astfel orice interferență între semnalele directe și cele reflectate. În ultimii ani, această tehnică a fost aplicată comercial – de exemplu, pentru 802.11 și pentru Bluetooth.

Ca o curiozitate, tehnica a fost co-inventată de Hedy Lamarr, o senzuală actriță născută în Austria, prima femeie care a apărut goală intr-un film (în 1933, în filmul cehesc Extase). Primul ei soț era fabricant de armament, și i-a spus cât de ușor era să blochezi undele radio, folosite pe atunci

pentru ghidarea torpilelor. Îngrozită când a descoperit că el îi vindea arme lui Hitler, ea s-a deghizat în servitoare pentru a fugi și a zburat la Hollywood pentru a-și continua cariera de actriță de film. În timpul liber, a inventat metoda salturilor de frecvențe pentru a-i ajuta pe aliați în război. Schema ei folosea 88 de frecvențe, numărul de clape (si frecvențe) de la pian. Pentru invenția lor, ea și prietenul ei, compozitorul de musical-uri George Antheil, au primit patentul U.S. cu numărul 2.292.387 . Totuși, nu au reușit sa convingă marina americană că invenția lor putea fi pusă în practică și nu au primit niciodată nici un fel de onoruri. Abia peste ani, după ce patentul expirase, metoda a devenit o tehnică populară.

Și cealaltă formă de spectru larg, **spectru larg cu succesiune directă** (**direct sequence spread spectrum**), metodă care împrăștie semnalul pe o bandă de frecvență largă câștigă popularitate în lumea comercială. In particular, unele telefoane mobile din cea de-a doua generație folosesc această tehnică, și va deveni predominantă pentru generația a treia, datorită eficienței spectrale bune, imunității la zgomot și a altor proprietăți. Este folosită și în unele LAN-uri fără fir. Vom reveni la discuția despre spectrul larg mai târziu în acest capitol. Pentru o istorie fascinantă și detaliată a comunicațiilor în spectru larg, vezi (Scholtz, 1982).

Pentru moment, vom considera că toate transmisiunile folosesc o bandă de frecvență îngustă. Vom discuta despre modul în care diverse părți ale spectrului electromagnetic din fig. 2-11 sunt folosite, începând cu banda radio.

2.3.2 Transmisia radio

Undele radio sunt ușor de generat, pot parcurge distanțe mari, penetrează cu ușurință zidurile clădirilor, fiind larg răspândite în comunicații, atât interioare cât și exterioare. De asemenea, undele radio sunt omnidirecționale, ceea ce înseamnă că se pot propaga în orice direcție de la sursă, deci nu este nevoie de o aliniere fizică a transmițătorului cu receptorul.

Uneori această proprietate de propagare omnidirecțională este bună, alteori nu. În anii ‘70, General Motors a decis să echipeze noile sale Cadillac-uri cu un calculator care să prevină blocarea frânelor. Atunci când șoferul apăsa pedala de frână, calculatorul frâna treptat, în loc să preseze frâna complet. Într-o zi frumoasă de vară, un ofițer de pe o autostradă din Ohio a început să-și folosească stația radio mobilă pentru a chema sediul central și deodată un Cadillac situat în apropriere a început să se cabreze ca un cal sălbatic. Atunci când ofițerul a oprit mașina, șoferul a pretins că el nu a făcut nimic, dar mașina a înnebunit.

În cele din urmă a reieșit că lucrurile se petreceau după un anumit tipar: mașinile Cadillac erau scăpate uneori de sub control, dar numai pe marile autostrăzi din Ohio și numai când o patrulă de poliție era în zonă. Pentru o foarte lungă perioadă de timp, cei de la General Motors nu au înțeles de ce mașinile mergeau foarte bine în toate celelalte state, ca și pe străzile secundare din Ohio. După îndelungi căutări au descoperit că, în Cadillac, cablajul forma o antenă foarte bună pentru frecvența folosită de noul sistem radio al poliției rutiere din Ohio.

Proprietățile undelor radio sunt dependente de frecvență. La frecvențe joase, undele radio se propagă bine prin obstacole, dar puterea semnalului scade mult odată cu distanța de la sursă, aproximativ cu $1/r^2$ în aer. La frecvențe înalte, undele radio tind să se propage în linie dreaptă și să ricoșeze din obstacole. De asemenea, sunt absorbite de ploaie. Mai mult, toate frecvențele radio sunt supuse la interferențe datorate motoarelor și altor echipamente electrice.

Datorită capacității undelor radio de a se propaga pe distanțe mari, interferența dintre utilizatori devine o problemă. Acesta este principalul motiv pentru care toate guvernele acordă cu foarte mare atenție licențele pentru utilizatorii de transmițătoare radio, cu o singură excepție (discutată mai jos).

În benzile de frecvență foarte joasă (VLF), joasă (LF) și medie (MF), undele radio se propagă la sol, după cum este ilustrat în fig. 2-12(a). Aceste unde pot fi detectate pe distanțe de până la aproximativ 1000 Km pentru frecvențele mai joase și pe distanțe mai mici pentru cele mai înalte. Pentru difuzarea undelor radio AM se folosește banda MF, acesta fiind motivul pentru care stația radio AM din Boston nu poate fi auzită cu ușurință în New York. Undele radio în această bandă trec ușor prin clădiri, fiind astfel posibilă utilizarea radiourilor portabile în spații interioare. Problema principală care apare la comunicația de date la aceste frecvențe este lărgimea mică a benzii pe care o oferă [vezi ecuația (2-2)].

În benzile de frecvență înaltă și foarte înaltă, undele de la sol tind să fie absorbite de pământ. Totuși, undele care ating ionosfera (un strat de particule care învelește atmosfera la o înălțime de 100 până la 500 Km), sunt refractate de aceasta și trimise înapoi spre pământ, după cum arată fig. 2-12(b). În anumite condiții atmosferice, semnalele pot parcurge acest drum de mai multe ori.

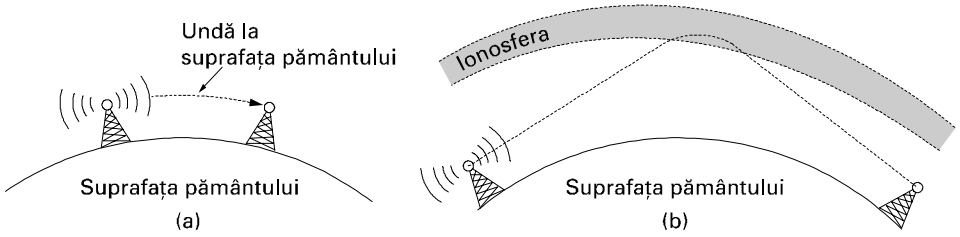


Fig. 2-12. (a) În benzile VLF, VF și MF, undele radio urmăresc curbura pământului. (b) În banda HF undele revin din ionosferă.

Operatorii radio amatori folosesc aceste benzi pentru a realiza convorbiri la mare distanță. De asemenea, comunicațiile armatei se desfășoară în benzile de frecvențe înalte și foarte înalte.

2.3.3 Transmisia prin microunde

Peste 100 MHz, undele se propagă în linii aproximativ drepte și pot fi, din acest motiv, direcționate. Concentrând toată energia într-un fascicol îngust, cu ajutorul unei antene parabolice (ca o antenă de satelit obișnuită) rezultă o valoare mult mai ridicată a raportului semnal-zgomot, dar antenele care transmit și cele care recepționează trebuie să fie aliniate cu precizie una cu alta. În plus, faptul că aceste unde sunt orientate permite ca mai multe transmițătoare să fie aliniate și să comunice cu mai multe receptoare aflate în linie, fără interferențe, cu condiția să se respecte câteva reguli de distanțare. Înaintea fibrelor optice, microundele au format, timp de decenii, inima sistemului telefonic de comunicație pe distanțe mari. De fapt, MCI, una dintre primele competitoare AT&T , și-a construit întregul sistem prin comunicații cu microunde, din turn în turn, la distanțe de zeci de kilometri între ele. Chiar și numele companiei reflectă acest lucru (MCI înseamnă Microwave Communications, Inc.). MCI a trecut apoi pe fibră și a fuzionat cu WorldCom.

Datorită faptului că microundele se propagă în linii drepte, dacă turnurile sunt foarte depărtate, atunci în calea lor stă chiar Pământul (gândiți-vă la o legătură între San Francisco și Amsterdam). În consecință trebuie instalate repetoare din loc în loc. Cu cât turnurile sunt mai înalte, cu atât distanța

dintre repetoare este mai mare. Distanța dintre repetoare crește direct proporțional cu radicalul înălțimii turnului. Pentru turnuri cu o înălțime de 100 m, distanța dintre repetoare poate fi de 80 km.

Spre deosebire de undele radio de frecvențe joase, microundele nu trec cu ușurință prin zidurile clădirilor. În plus, cu toate că unda poate fi bine direcționată la transmițător, apare o divergență în spațiu. Unele unde pot fi refractate de straturile atmosferice joase și pot întârzia mai mult decât undele directe. Undele întârziate pot sosi defazate față de unda directă, anulând astfel semnalul. Acest efect este numit **atenuare multi-căi** (**multipath fading**) și constituie deseori o problemă serioasă. Este dependentă de vreme și de frecvență. Unii operatori păstrează nefolosit un procent de 10 la sută din canalul propriu pentru a putea comuta pe acesta atunci când atenuarea multi-căi le anulează temporar anumite benzi de frecvență.

Cererea de benzi este din ce în ce mai mare și duce operatorii către frecvențe tot mai înalte. Benzi de până la 10 GHz sunt acum uzuale, dar la aproape 8 GHz apare o nouă problemă: absorbția de către apă. Aceste unde au lungimi de doar câțiva centimetri și sunt absorbite de ploaie. Acest efect ar fi potrivit pentru cineva care ar fi încercat să construiască un imens cuptor cu microunde în aer liber, dar pentru comunicații este o problemă dificilă. La fel ca și în cazul atenuării multi-căi, singura soluție posibilă este întreruperea legăturilor acolo unde plouă și găsirea unei rute alternative.

Comunicațiile cu microunde sunt atât de larg folosite de telefonia pe distanțe mari, telefoanele celulare, televiziune și altele, încât a apărut o criză în ceea ce privește spectrul. Microundele au mai multe avantaje semnificative față de fibră. Cel mai important avantaj este că nu sunt necesare drepturi de acces la drum, cumpărând un mic teren la fiecare 50 Km și montând un turn pe el, se poate ocoli sistemul telefonic și se poate realiza o comunicare directă. Astfel a reușit MCI să pornească atât de rapid ca o companie de telefoane pe distanțe mari. (Sprint a aplicat o altă tactică : a fost formată de Southern Pacific Railroad (căile feroviare sudice), care deja deținea destule drepturi de acces și tot ce a avut de făcut a fost să îngroape fibra lângă șine).

Comunicațiile cu microunde, prin comparație cu alte medii de transmisie, sunt ieftine. Prețul ridicării a două turnuri simple (doi stâlpi înalți asigurați cu patru cabluri) și de montare a unei antene pe fiecare turn, poate fi mai mic decât prețul îngropării a 50 de Km de fibră într-o zonă urbană foarte populată sau peste un munte și poate fi mai mic decât costul închirierii fibrei de la o companie telefonică, mai ales atunci când acestea nu au plătit încă integral cuprul care a fost înlocuit cu fibră.

Politica din domeniul spectrului electromagnetic

Pentru a preveni haosul general, există convenții naționale și internaționale care reglementează modul de folosire al frecvențelor – cine ce frecvență folosește. Deoarece toată lumea dorește o rată de transfer cât mai mare, toată lumea dorește cât mai mult din spectru. Guvernele naționale alocă spectru pentru radio AM și FM, televiziune, telefoane mobile, ca de altfel și pentru companii telefonice, poliție, marină, navigatori, armată, guvern și mulți alți clienți competitori. La nivel internațional agenția ITU-R (WARC) încearcă să coordoneze această alocare, astfel încât să poată fi construite dispozitive care să funcționeze în diverse țări. Totuși, țările nu sunt obligate să respecte recomandările ITU, iar FCC (Comisia Federală de Comunicații), comisia care se ocupă cu alocarea de spectru în Statele Unite, a respins de câteva ori recomandările ITU (de obicei pentru că acestea cereau unui grup puternic din punct de vedere politic să renunțe la o parte din spectru).

Chiar și atunci când o parte din spectru a fost alocată unei anumite utilizări, cum ar fi telefonia mobilă, apare discuția legată de modul de alocare a frecvențelor către companiile de telecomunicații. În trecut, erau utilizați trei algoritmi. Cel mai vechi algoritm, adesea denumit **concursul de frumusețe**, cerea fiecărei companii de telecomunicații să explice de ce propunerea sa servește cel mai

bine interesul public. Oficialii guvernului decideau apoi care dintre povești le place cel mai mult. Posibilitatea ca unul dintre oficialii să premieze compania favorită cu o proprietate valorând miliarde de dolari ducea adesea la mită, corupție, nepotism și chiar mai rău. Mai mult, chiar și un oficial cinstit al guvernului, căruia i se părea că o companie străină ar face o treabă mai bună decât companiile naționale, ar fi avut de dat multe explicații.

Această observație a condus la algoritmul 2: organizarea unei loterii între companiile interesate. În acest caz, problema este că pot participa la loterie chiar și companii care nu au nici o intenție de utilizare a spectrului. Să zicem că un restaurant cu servire rapidă sau un lanț de magazine de pantofi ar câștiga: acesta ar putea revinde spectrul unei companii de telecomunicații, cu un profit imens și fără riscuri.

Scandaluri furtunoase datorate unor astfel de situații alarmante, deși aleatoare, au dus la critici severe din partea multora, ceea ce a condus la algoritmul 3: licitarea benzii de frecvență și atribuirea ei celui care dă mai mult. În anul 2000, când Anglia a licitat frecvențele necesare pentru generația a treia de sisteme mobile, se așteptau să primească în jur de 4 miliarde de dolari. De fapt, au primit 40 de miliarde de dolari, deoarece companiile de telecomunicații au intrat într-o frenezie molipsitoare, fiind speriate de moarte că vor rata trenul telefoniei mobile. Acest eveniment a declanșat lăcomia guvernelor din jur și le-a inspirat să organizeze propriile licitații. A funcționat, dar a și lăsat unele dintre companiile de telecomunicații cu foarte multe datorii, aproape de faliment. Chiar și în cele mai bun variante, vor trece mulți ani până când acestea vor reuși să acopere taxele de licență.

O modalitate cu totul diferită de a aborda problema alocării frecvențelor este să nu fie alocate în nici un fel. Toată lumea este lăsată să transmită cât dorește, fiind reglementată doar puterea folosită, astfel încât stațiile să aibă o rază de acțiune suficient de scurtă ca să nu interfereze între ele. În consecință, cele mai multe guverne au pus deoparte câteva benzi de frecvență, numite benzi **ISM (Industriale, Științifice, Medicale)** pentru utilizare nelicențiată. Sistemele de deschidere de uși de garaj, telefoane fără fir, jucării telecomandate, mouse-uri fără fir, și multe alte aparate casnice fără fir folosesc benzile ISM. Pentru a minimiza interferența între aceste dispozitive necoordonate, FCC a impus ca toate dispozitivele din benzile ISM să folosească tehnici de spectru larg. Reguli asemănătoare sunt aplicate și în alte țări.

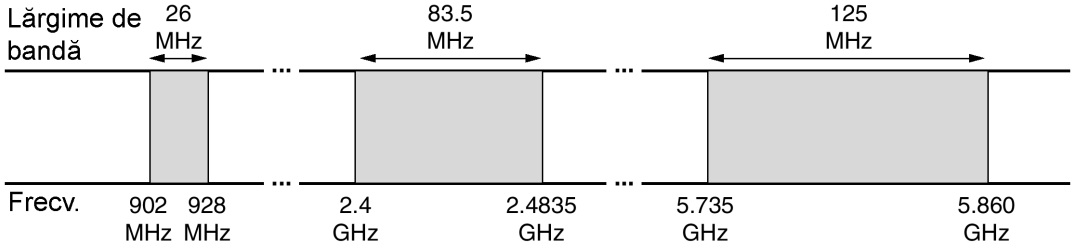


Fig. 2-13. Benzile ISM în Statele Unite.

Localizarea benzilor ISM diferă oarecum de la o țară la alta. În Statele Unite, de exemplu, dispozitivele a căror putere este sub 1 Watt pot folosi benzile din fig. 2.13 fără a cere licență FCC. Banda 900Mhz este cea mai bună, dar este aglomerată și nu este disponibilă în întreaga lume. Banda 2.4GHz este disponibilă în cele mai multe țări, dar este supusă interferenței cu instalațiile radar sau cuptoarele cu microunde. Bluetooth și unele dintre LAN-urile fără fir conforme 802.11 operează pe această bandă. Banda 5.7GHz este nouă și relativ nedezvoltată, astfel că echipamentele pentru ea sunt scumpe; dar pentru că 802.11 o folosește, va deveni rapid mai populară.

2.3.4 Undele infraroșii și milimetrice

Undele infraroșii și milimetrice neghidate sunt larg folosite pentru comunicațiile pe distanțe mici. Telecomenzile pentru televizoare, aparate video sau combine muzicale folosesc comunicațiile în infraroșu. Ele sunt relativ direcționale, ieftine și ușor de construit, dar au un dezavantaj major: nu penetrează obiectele solide (încercați să stați între telecomandă și televizor și vedeți dacă telecomanda mai dă comenzi televizorului). În general, pe măsură ce ne deplasăm de la undele radio lungi către lumina vizibilă, undele se comportă din ce în ce mai mult ca lumină și din ce în ce mai puțin ca unde radio.

Pe de altă parte, faptul că razele infraroșii nu trec prin obiecte constituie un avantaj. Aceasta înseamnă că un sistem cu infraroșii dintr-o cameră a unei clădiri nu va interfera cu un sistem similar situat în camerele sau clădirile adiacente: nu poți controla televizorul vecinului tău cu telecomanda ta. Mai mult, sistemele de protecție cu radiații infraroșii asigură o mai bună protecție împotriva interceptărilor în raport cu sistemele radiofonice, tocmai din acest motiv. Datorită acestor motive, pentru operarea unui sistem cu infraroșii nu este necesară procurarea unei licențe, spre deosebire de sistemele radiofonice, care trebuie să dețină o licență.

2.3.5 Transmisia undelor luminoase

Semnalele optice neghidate au fost folosite secole întregi. Înaintea faimoasei lui călătorii, Paul Revere a folosit semnalizarea optică binară de la Old North Church. O aplicație mai modernă este conectarea rețelei locale între două clădiri prin intermediul laserelor montate pe acoperișurile acestora. Semnalizarea optică folosind laserul este inerent unidirecțională, deci fiecare clădire are nevoie de propriul ei laser și de propria ei fotodiodă. Această schemă oferă o bandă foarte largă la un cost foarte redus. De asemenea, este ușor de instalat și, spre deosebire de microunde, nu necesită o licență FCC. Puterea laserului, un fascicol foarte îngust, este aici o slăbiciune. Îndreptarea unui fascicol de lumină de 1mm lățime către o țintă de lățimea unui ac cu gămălie aflată la 500 de metri depărtare necesită o tehnică de vârf. De obicei sunt folosite lentile pentru o ușoară defocalizare a fascicolului.

Un alt dezavantaj este că fascicolul laser nu penetrează ploaia și ceața groasă, dar în mod normal ele funcționează bine în zilele însorite. Totuși, autorul a participat odată într-un hotel modern din Europa la o conferință la care organizatorii conferinței s-au gândit să pună la dispoziție o cameră plină cu terminale, în care participanții să-și poată citi poșta electronică în timpul prezentărilor plictisitoare. Deoarece PTT-ul local nu dorea să instaleze un număr mare de linii telefonice doar pentru 3 zile, organizatorii au montat pe acoperiș un laser orientat către clădirea departamentului de calculatoare al universității de calculatoare aflată la o distanța de câțiva kilometri. Ei l-au testat cu o noapte înainte și totul a decurs perfect. În dimineața următoare, într-o zi însorită, la ora 9, legătura a căzut și a rămas nefuncțională toata ziua. Seara, organizatorii au testat-o din nou cu atenție și a funcționat încă o dată perfect. Același lucru s-a întâmplat încă două zile la rând.

După conferință, organizatorii au descoperit problema. Căldura datorată soarelui din timpul zilei a determinat nașterea unor curenți de convecție din acoperișul clădirii, ca în fig. 2-14. Acest aer turbulent a deviat fascicolul și l-a făcut să oscileze în jurul detectorului. Această „vedere” atmosferică face ca stelele să pâlpâie (acesta este motivul pentru care astronomii își pun telescoapele pe vârful munților – ca să fie cât se poate de mult deasupra atmosferei). Efectul respectiv este responsabil și pentru „tremurul” șoselei într-o zi însorită și a imaginii vălurite deasupra unui radiator fierbinte.

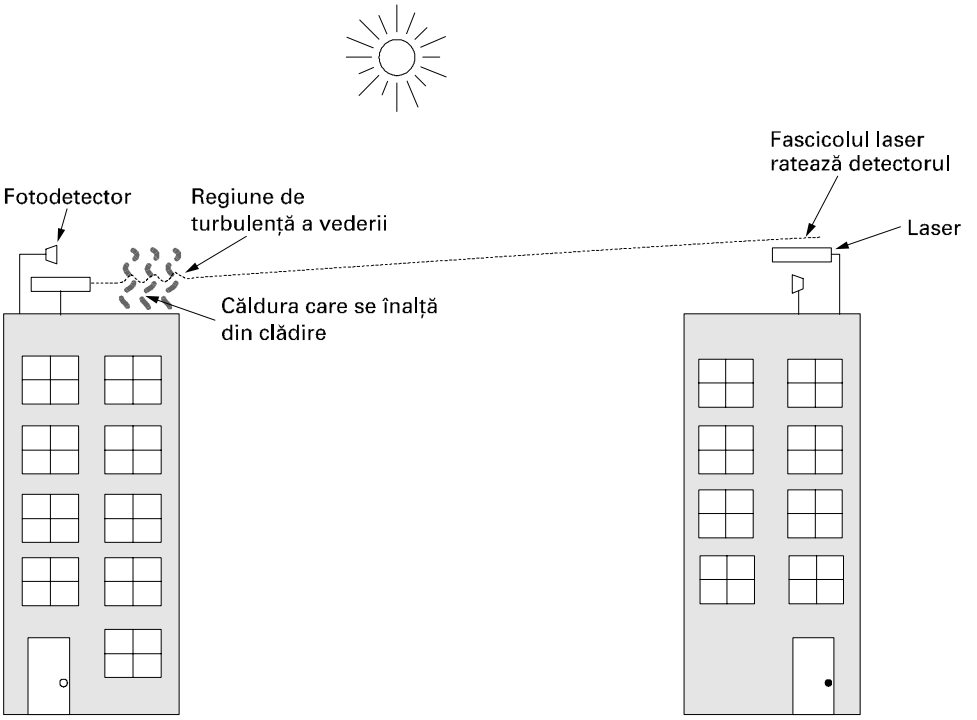


Fig. 2-14. Curenții de convecție pot interfera cu sistemele de comunicație prin laser. Aici este prezentat un sistem bidirecțional, cu două lasere.

2.4 SATELIȚI DE COMUNICAȚIE

În anii 1950 și la începutul anilor 1960, oamenii au încercat să construiască sisteme de comunicație pe baza reflectării semnalelor de către baloanele meteorologice metalizate. Din nefericire, semnalele recepționate erau prea slabe ca să poată fi folosite practic la ceva. Apoi, Marina S.U.A. a observat pe cer un fel de balon meteorologic permanent - Luna - și a construit, pe baza reflectării semnalelor de către Lună, un sistem operațional pentru comunicații navă-țărm.

Progresul în domeniul comunicațiilor celeste a trebuit să mai aștepte până când a fost lansat primul satelit de comunicații, în 1962. Principala diferență între un satelit artificial și unul natural este aceea că satelitul artificial poate amplifica semnalele înainte de a le transmite înapoi, transformând ceea ce era doar o curiozitate într-un sistem puternic de comunicație.

Sateliții de comunicație au câteva proprietăți interesante, care îi fac tentanți pentru multe aplicații. Un satelit de comunicație poate fi gândit ca un mare repetor de microunde, aflat în cer. Acesta conține mai multe **dispozitive de recepție-transmisie automată** (**transporder**); fiecare dintre ele ascultă pe o anume porțiune din spectru, amplifică semnalul recepționat și apoi îl redifuzează pe o altă frecvență, pentru a evita interferența cu semnalul recepționat. Unda descendentă poate fi difuzată, acoperind astfel o fracțiune substanțială din suprafața Pământului sau poate fi concentrată, caz în

care va acoperi numai o zonă de câteva sute de kilometri în diametru. Acest mod de operare este cunoscut și sub numele de **țeavă îndoită** (**bent pipe**).

Conform legii lui Kepler, perioada de rotație a unui satelit variază cu puterea 3/2 a razei orbitei. Cu cât un satelit este mai sus, cu atât este mai lungă perioada. În apropierea suprafeței Pământului, perioada este de aproximativ 90 de minute. În consecință, sateliții cu orbita mai mică dispar din raza vizuală destul de repede, astfel că mulți dintre ei sunt necesari pentru a oferi acoperire continuă. La o altitudine de aproximativ 35.800 km, perioada unui satelit este de 24 de ore. La o altitudine de aproximativ 384.000 km, perioada unui satelit este de aproape o lună, așa după cum poate confirma oricine care a urmărit luna în mod regulat.

Perioada unui satelit este importantă, dar nu este singurul criteriu în determinarea locului în care va fi plasat satelitul. Un alt criteriu este prezența centurilor lui Van Allen, straturi de particule foarte încărcate prinse în câmpul magnetic al pământului. Acești factori conduc la formarea a trei regiuni în care sateliții pot fi plasați în siguranță. Aceste regiuni și unele dintre proprietățile lor sunt ilustrate în fig. 2-15. În continuare vom descrie pe scurt sateliții care sunt plasați în fiecare dintre aceste regiuni.

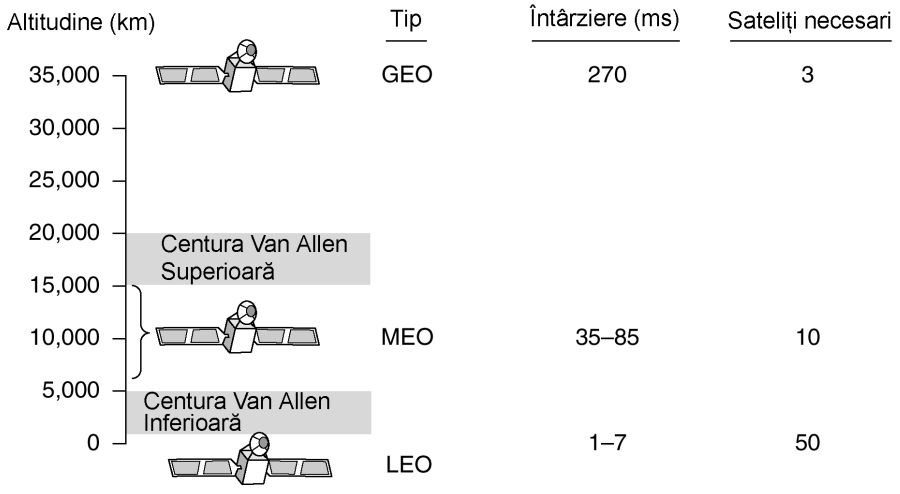


Fig. 2-15. Sateliții de comunicație și câteva dintre proprietățile lor, inclusiv altitudinea, întârzierea pentru un drum dus-întors și numărul de sateliți necesari pentru acoperirea globală.

2.4.1 Sateliți geostaționari

În 1945, scriitorul de proză științifico-fantastică Arthur C. Clarke a calculat că un satelit aflat la altitudinea de 35.800 km pe o orbită ecuatorială circulară apare ca staționar pe cer, astfel încât nu este nevoie să fie urmărit (Clarke, 1945). El a mers mai departe și a descris un sistem complet de comunicare care folosea acești **sateliți geostaționari**, inclusiv orbitele lor, panourile solare, frecvențele radio și procedurile de lansare.

Din păcate, a ajuns la concluzia că sateliții nu erau o soluție practică datorită imposibilității de a lansa pe orbită amplificatoare cu tub catodic, acestea fiind fragile și mari consumatoare de energie, așa că nu a mai continuat pe aceasta idee, deși a scris câteva povestiri științifico-fantastice bazate pe ea.

Inventarea tranzistorului a schimbat toate acestea și a fost lansat primul satelit artificial pentru comunicații, Telstar, în iulie 1962. Din acel moment, sateliții pentru comunicații au devenit o afacere de miliarde de dolari și totodată singurul aspect al explorării spațiului cosmic care a devenit foarte profitabil. Acești sateliți care zboară la altitudini mari sunt adeseori numiți sateliți **GEO** (**Geostationary Earth Orbit**, rom: cu orbită geostaționară terestră).

Pentru a evita interferența, în condițiile tehnologiilor actuale, nu este recomandat să existe sateliți mai apropiați de 2 grade în planul ecuatorial de 360 grade. La o spațiere de 2 grade, pot exista pe cer, la un moment dat, doar 360/2=180 sateliți de comunicație geostaționari. Totuși, fiecare transponder poate folosi mai multe frecvențe și polarizări pentru a crește lungimea de bandă disponibilă.

Pentru a preveni un haos total pe cer, alocarea locurilor pe orbită este făcută de către ITU. Acest proces este mult influențat de politică, existând țări de abia ieșite din epoca de piatra care își cer locurile „lor" pe orbită (pentru a le închiria celui care oferă mai mult). Alte țări susțin că drepturile naționale de proprietate nu se extind până la lună, precum și că nici o țară nu are dreptul să revendice locurile de pe orbita de deasupra teritoriului său. Pentru a pune paie pe foc, telecomunicațiile comerciale nu reprezintă singurul domeniu interesat de acești sateliți. Canalele de televiziune, guvernele și armata doresc și ele câte o felie din „plăcinta" reprezentată de orbita terestra.

Sateliții moderni pot fi destul de mari, cântărind până la 4000 de kg și consumând câțiva KW de energie electrică produsă de panourile solare. Efectele forțelor de gravitație solare, lunare și planetare tind să îi deplaseze din locurile și orientările prevăzute pentru ei pe orbită, efect contracarat de motoarele de rachetă cu care sunt echipați. Această activitate de reglare fină se numește **menținerea stației**. Cu toate acestea, când combustibilul pentru motoare se epuizează, de obicei în vreo 10 ani, satelitul plutește în derivă și se rotește neajutorat, astfel încât trebuie să fie scos din funcționare. În cele din urma, orbita sa se deformează și satelitul reintră în atmosfera unde este distrus prin ardere sau, uneori, se prăbușește pe pământ.

Alocarea locurilor pe orbită nu este singurul măr al discordiei. Alocarea frecvențelor este de asemenea un motiv de a se isca noi conflicte, deoarece transmisiile descendente de la sateliți interferează cu utilizatorii dispozitivelor de comunicație prin microunde deja existente. Prin urmare, ITU a alocat anumite benzi de frecvență utilizatorilor comunicațiilor prin sateliți. Cele mai importante benzi comerciale sunt listate în fig. 2-16. Banda C a fost desemnată inițial pentru traficul comercial prin sateliți. În banda C sunt asigurate două domenii de frecvență, cea mai mică pentru traficul descendent (dinspre satelit), iar cea superioară pentru traficul ascendent (către satelit). Pentru o conexiune full-duplex este necesar un canal în ambele sensuri. Aceste benzi sunt deja supraaglomerate, deoarece sunt folosite și de purtătoarele obișnuite pentru legăturile terestre pe microunde. Benzile L și S au fost adăugate prin acordul internațional din anul 2000. Oricum, și ele sunt înguste și aglomerate.

Banda	Legătura descendentă	Legătura ascendenta	Lățime de bandă	Probleme
L	1.5 GHz	1.6 GHz	15 MHz	Lățime mică de bandă; aglomerata
S	1.9 GHz	2.2 GHz	70 MHz	Lățime mică de bandă; aglomerata
C	4.0 GHz	6.0 GHz	500 MHz	Interferențe terestre
Ku	11 GHz	14 GHz	500 MHz	Ploaia
Ka	20 GHz	30 GHz	3500 MHz	Ploaia; costul echipamentelor

Fig. 2-16. Principalele benzi de satelit.

Următoarea bandă, cea mai înaltă, disponibilă pentru companiile comerciale de telecomunicații, este banda Ku (K under, rom: sub K). Această bandă nu este (încă) congestionată și – la aceste frecvențe – sateliții pot fi poziționați la o distanță de un grad. Totuși, există și aici o altă problemă: ploaia. Apa este un absorbant excelent al acestor microunde scurte. Din fericire, furtunile torențiale sunt de obicei localizate și, prin urmare, folosind mai multe stații terestre separate de distanțe mari (în loc de una singură), problema poate fi evitată cu prețul surplusului de antene, cabluri și electronică necesare pentru a comuta rapid între stații. Lățimea de bandă pentru Ka (K above, rom: peste K) a fost și ea alocată pentru traficul comercial prin satelit, dar echipamentele necesare pentru folosirea ei sunt încă foarte scumpe. În plus față de aceste benzi comerciale, există multe benzi guvernamentale și militare.

Un satelit modern are în jur de 40 de transpondere, fiecare cu o lățime de bandă de 80 MHz. Un transponder de 50 Mbps poate fi folosit pentru a codifica un singur flux de date de 50 Mbps, 800 canale vocale digitale de 64 Kbps sau diverse alte combinații. De obicei, fiecare transponder funcționează ca un repetor, dar, mai nou, sateliții au și o anumita capacitate de procesare integrată, permițând operații mai sofisticate. La sateliții mai vechi, împărțirea transponderilor pe canale s-a făcut static, prin împărțirea lărgimii de bandă în benzi fixe de frecvență (FDM). În prezent, fiecare fascicol de transponder este împărțit în intervale de timp, cu mai mulți utilizatori comunicând pe rând. Vom studia aceste două tehnici (multiplexarea cu divizare în frecvență și multiplexarea cu divizare în timp) într-o secțiune următoare din acest capitol.

Primii sateliți geostaționari aveau un singur fascicol spațial care ilumina aproximativ 1/3 din suprafața Pământului, suprafață numită **rază de acțiune**. Odată cu scăderea masivă a prețului, dimensiunii și cerințelor de putere ale microelectronicii, a devenit posibilă o strategie de difuzare mai complexă. Fiecare satelit este echipat cu antene și transpondere multiple. Fiecare fascicol descendent poate fi focalizat pe o arie geografică mică și prin urmare pot avea loc simultan, transmisii ascendente și descendente multiple. Aceste așa numite **fascicole punctuale** sunt în mod obișnuit de formă eliptică și pot avea până la câteva sute de km în diametru. Un satelit de comunicații pentru Statele Unite are în mod normal un singur fascicol larg pentru cele 48 de state alăturate, plus fascicole punctuale pentru Alaska și Hawaii.

O nouă realizare în lumea comunicațiilor prin satelit o constituie dezvoltarea microstațiilor de cost scăzut, denumite și **VSAT**-uri (**Very Small Aperture Terminals**, rom: terminale cu deschidere foarte mică) (Abramson, 2000). Aceste mici terminale au antene de 1 metru sau mai mici (fata de cei 10m ai unei antene **GEO** standard) și pot emite cu o putere de aproximativ 1Watt. Legătura ascendentă este în general bună pentru 19.2Kbps, dar cea descendentă este mai mare, deseori de 512 Kbps. Televiziunea cu difuzare directă prin satelit folosește această tehnologie pentru transmisia uni-direcțională.

În multe sisteme VSAT, microstațiile nu au suficientă putere pentru a comunica direct între ele (prin intermediul satelitului, desigur). În schimb, se folosește o stație terestră specială, un hub, cu o antenă mare, de câștig ridicat, pentru a retransmite traficul dintre VSAT-uri, așa cum este prezentat în fig. 2-17. În acest fel, atât emițătorul cât și receptorul dispun de o antenă mare și de un amplificator puternic. Compromisul constă într-o întârziere mai mare în schimbul unor stații mai ieftine la utilizatorul final.

VSAT-urile au un potențial ridicat în special pentru zonele rurale. Nu este un lucru foarte cunoscut, dar peste jumătate din populația globului locuiește la peste o ora de mers pe jos de cel mai apropiat post telefonic. A instala fire telefonice până la fiecare dintre miile de sate mici este mult peste bugetele majorității guvernelor din lumea a treia, dar instalarea de antene VSAT de 1 metru alimentate de celule solare este o soluție adeseori posibilă. VSAT-urile oferă o tehnologie care va conecta întreaga lume.

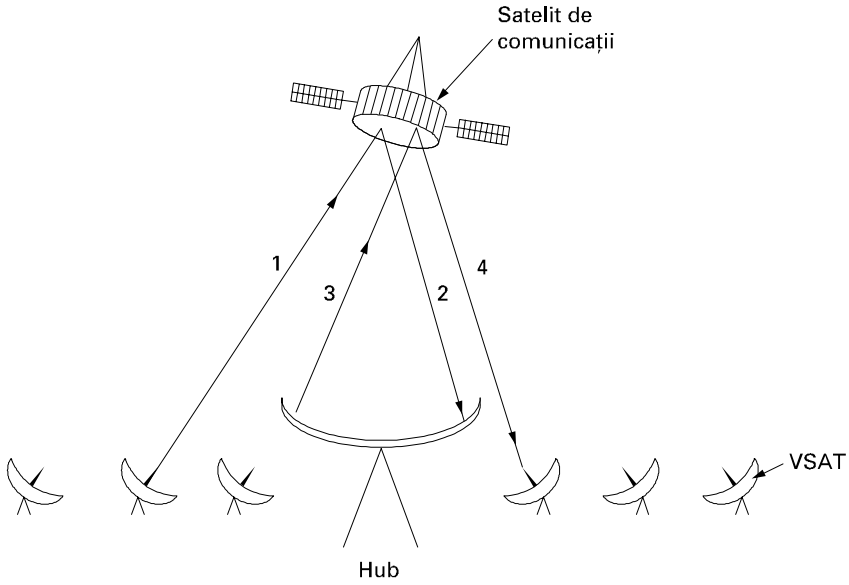


Fig. 2-17. VSAT-uri care folosesc un hub.

Sateliții de comunicație au câteva proprietăți prin care se deosebesc radical de legăturile terestre punct-la-punct. Ca un prim aspect, cu toate că semnalele spre și dinspre satelit se propagă cu viteza luminii (aproximativ 300.000 km/sec), lungimea semnificativă a traseului dus-întors introduce o întârziere substanțială pentru sateliții **GEO**. Funcție de distanța dintre utilizator și stația terestră, precum și de înălțimea satelitului deasupra orizontului, timpul de propagare este între 250 și 300 msec. O valoare uzuală este de 270 msec (540 msec pentru un sistem VSAT cu un hub).

Trebuie spus, pentru comparație, că legăturile terestre prin microunde au o întârziere de propagare în jur de 3 μsec/km, iar legăturile pe cablu coaxial sau fibră optică au o întârziere de aproximativ 5μsec/km. Transmisia prin cel de-al doilea mediu este mai înceată decât prin primul deoarece semnalele electromagnetice se propagă mai repede prin aer decât prin materiale solide.

O altă proprietate importantă a sateliților este aceea că ei sunt în mod inerent sisteme cu difuzare. Transmiterea unui mesaj către miile de stații din raza de acțiune a unui transponder costă tot atât de mult cât pentru o singură stație. Pentru unele aplicații, această proprietate este foarte utilă. De exemplu, se poate imagina un satelit difuzând pagini Web populare către memoriile cache ale unui mare număr de computere răspândite pe o arie largă. Chiar și atunci când difuzarea poate fi simulată folosind linii punct-la-punct, difuzarea prin satelit poate fi mult mai ieftină. Pe de altă parte, din punct de vedere al securității și confidențialității, sateliții sunt un dezastru complet: oricine poate asculta orice. Criptarea este esențială dacă securitatea este o necesitate.

Sateliții au și proprietatea că prețul transmisiei unui mesaj este independent de distanța parcursă. Un apel peste ocean nu costă mai mult decât un apel peste stradă. Sateliții au rate de eroare foarte scăzute și pot fi instalați aproape instantaneu, un considerent major pentru comunicațiile militare.

2.4.2 Sateliți de altitudine medie

La altitudini mult mai joase, între cele două centuri Van Allen, găsim sateliții MEO (Medium-Earth Orbit, rom: orbită terestră medie). Văzuți de pe Pământ, acești sateliți se deplasează lent în plan longitudinal, iar un ocol în jurul Pământului durează cam de 6 ore. În consecință, ei trebuie urmăriți în timp ce trec pe deasupra Pământului. Având altitudine mai mică decât **GEO**, au o rază de acțiune mai mică pe Pământ și este nevoie de emițătoare mai puțin puternice pentru a comunica cu ei. Deocamdată nu sunt folosiți pentru comunicații, deci nu vor mai fi examinați aici. Cei 24 de sateliți **GPS** (**Global Positioning System**, rom: sistem de poziționare global) care orbitează la aproximativ 18.000 km sunt exemple de sateliți MEO.

2.4.3 Sateliți de joasă altitudine

Coborând în altitudine, ajungem la sateliții **LEO** (**Low-Earth Orbit**, rom: orbită terestră joasă). Datorită mișcării lor rapide, este nevoie de mulți astfel de sateliți pentru a realiza un sistem complet. Pe de altă parte, datorită faptului că sateliții sunt atât de aproape de Pământ, stațiile terestre nu au nevoie de multă putere, iar întârzierea dus-întors este numai de câteva milisecunde. În acest paragraf vom examina trei exemple, două concepute pentru comunicațiile de voce și unul pentru servicii Internet.

Iridium

Așa cum am menționat și mai sus, în primii 30 de ani ai erei sateliților, sateliții de joasă altitudine au fost rareori folosiți, deoarece ei apar și dispar destul de repede din câmpul vizual. În 1990, Motorola a declanșat o acțiune în acest domeniu prin punerea la punct a unei noi aplicații. Motorola a obținut acordul FCC în vederea lansării a 77 de sateliți de joasă altitudine pentru proiectul Iridium (elementul 77 este Iridium). Planul a fost mai târziu revăzut, astfel încât să se utilizeze numai 66 de sateliți și, ca urmare, proiectul ar fi trebuit să fie redenumit Dysprosium (elementul 66), dar probabil că sună prea mult ca o boală. Ideea era că în momentul în care un satelit dispare din câmpul vizual, un alt satelit ar putea să-i ia locul. Această propunere a generat printre celelalte companii de telefoane o poftă nebună: dintr-o dată, toată lumea dorea să lanseze un lanț de sateliți de joasă altitudine.

După șapte ani în care s-au adunat parteneri și finanțări, sateliții Iridium s-au lansat în 1997. Serviciile de comunicații au început în noiembrie 1998. Din păcate, cererea comercială pentru telefoane mari și grele, chiar dacă acestea comunicau prin satelit, era neglijabilă deoarece rețeaua de telefonie mobilă crescuse spectaculos din 1990. În consecință, Iridium nu era profitabil și a fost forțat să intre în faliment în august 1999, într-unul din cele mai spectaculoase fiasco-uri de corporații din istorie. În consecință, sateliții și celelalte bunuri (în valoare de 5 miliarde $) au fost cumpărate de un investitor cu 25 milioane $ într-un fel de vânzare cu reducere dintr-un garaj extraterestru. Serviciul Iridium a fost repornit în martie 2001.

Scopul principal al sistemului Iridium era (și este) să furnizeze servicii mondiale de telecomunicație, folosind dispozitive portabile care să comunice direct cu sateliții Iridium. Sistemul furnizează servicii vocale, de date, paging, fax și navigare, oriunde pe pământ, apă și aer. Printre clienți se numără industriile maritimă, aviatică și de explorare de petrol, ca și persoanele care călătoresc în părți ale lumii lipsite de o infrastructură de telecomunicații (de exemplu deșerturi, munți, jungle și unele țări din lumea a treia).

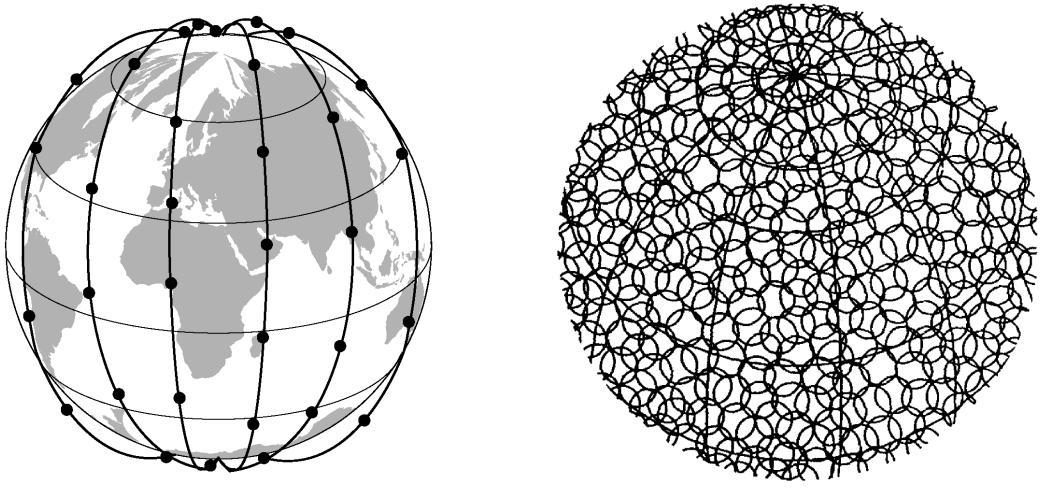


Fig. 2-18. (a) Sateliții Iridium formează șase coliere în jurul Pământului. (b) 1628 de celule mișcătoare acoperă Pământul.

Sateliții Iridium sunt poziționați la o altitudine de 750 km pe orbite polare circulare. Ei sunt aranjați în formă de coliere nord-sud, cu un satelit la fiecare 32 grade latitudine. După cum se sugerează în fig. 2-18(a), cu șase coliere de sateliți s-ar putea acoperi întregul Pământ. Cei care nu cunosc prea multe despre chimie, se pot gândi la această dispunere ca la un atom de dysprosium foarte mare, având Pământul pe post de nucleu și sateliții pe post de electroni.

După cum este prezentat în fig. 2-18(b), fiecare satelit va avea cel mult 48 de raze punctuale, cu un total de 1628 celule pe suprafața Pământului. Fiecare satelit are o capacitate de 3840 de canale, sau 253.440 în total. O parte din acestea sunt folosite pentru paging și navigare, altele pentru date și voce.

O proprietate interesantă a sistemului Iridium este aceea că o comunicație între clienți aflați la distanță are loc în spațiu, cu un satelit transmițând date la altul, cum este ilustrat în fig. 2-19(a). Aici putem vedea un apelant de la Polul Nord contactând un satelit aflat direct deasupra sa. Apelul este transmis prin ceilalți sateliți și trimis în cele din urma la apelatul de la Polul Sud.

Globalstar

Un proiect alternativ pentru Iridium este Globalstar. Acesta se bazează pe 48 de sateliți LEO, dar folosește o schemă de comutare diferită de cea folosită de Iridium. În timp ce Iridium transmite apeluri de la satelit la satelit, ceea ce necesita echipamente complicate de comutare în cadrul sateliților, Globalstar utilizează un sistem de repetoare tradițional. Apelul pornit de la Polul Nord din fig. 2-19(b) este trimis înapoi către pământ și preluat de stația terestră mare de la Atelierul lui Moș Crăciun. Apelul este apoi dirijat printr-o rețea terestră către stația terestră cea mai apropiată de apelat și livrat printr-o conexiune cu satelitul repetor, așa cum se vede și în figură. Avantajul acestei scheme este că transferă mare parte din complexitate pe Pământ, unde este mai prelucrările sunt mai ușor de efectuat. De asemenea, dacă pentru stațiile terestre se vor folosi antene mari, care pot emite un semnal puternic și pot recepționa unul slab, atunci în sistem pot fi folosite telefoane cu putere mică. Până la urmă, telefonul debitează o putere de câțiva mW, deci semnalul care se întoarce la stația terestră este destul de slab, chiar după ce a fost amplificat de satelit.

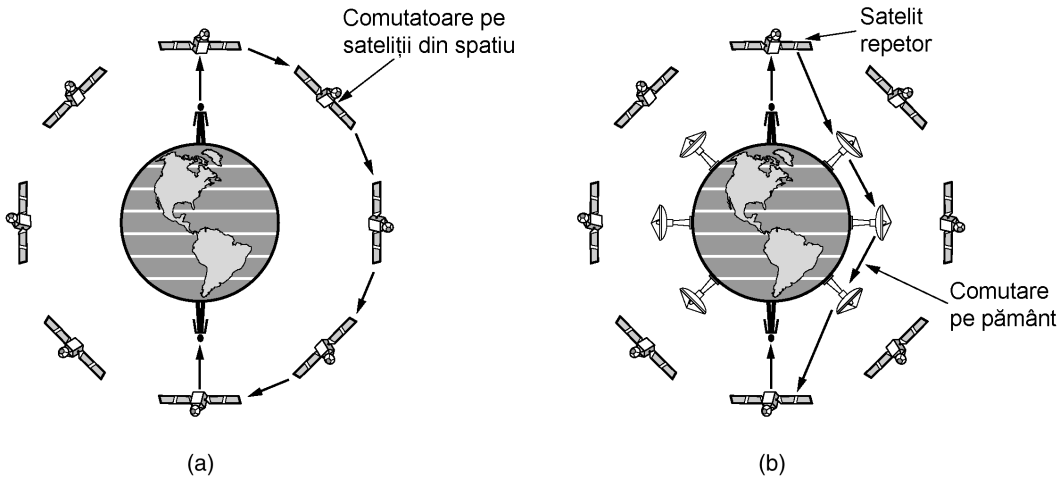


Fig. 2-19. (a) Transmisia în spațiu.
(b) Transmisia pe Pământ.

Teledesic

Iridium este conceput pentru utilizatorii de telefoane aflați în locuri ciudate. Următorul nostru exemplu, **Teledesic**, este conceput pentru utilizatorii de Internet din întreaga lume doritori de cât mai multă lărgime de bandă. A fost conceput în 1990 de către pionierul telefoniei mobile Craig McCaw și de către fondatorul Microsoft, Bill Gates, care era nemulțumit de viteza de melc cu care companiile telefonice din lume ofereau lățime ridicată de bandă utilizatorilor de calculatoare. Scopul sistemului Teledesic este să ofere milioanelor de utilizatori Internet care accesează sistemul simultan o legătură ascendentă de până la 100 Mbps și o legătură descendentă de până la 720 Mbps, utilizând o antenă de tip VSAT mică și fixă, dar și ocolind total sistemul telefonic. Pentru companiile telefonice, acesta este un ideal de neatins.

Proiectul original consta dintr-un sistem format din 288 de sateliți cu rază mică de acțiune, poziționați în 12 planuri imediat sub centura Van Allen, la o altitudine de 1350 km. Acest proiect a fost schimbat ulterior la 30 de sateliți cu raze de acțiune mai mari. Transmisia se petrece în banda Ka, relativ neaglomerata și cu lățime de bandă ridicată. Sistemul funcționează cu comutare de pachete în spațiu, fiecare satelit fiind capabil să dirijeze pachete către vecinii săi. Când un utilizator are nevoie de lățime de bandă pentru a trimite pachete, aceasta este cerută și alocată dinamic în aproximativ 50msec. Este prevăzut ca sistemul să intre în funcțiune în anul 2005, dacă totul merge conform planului.

2.4.4 Sateliții în comparație cu fibrele optice

O comparație între comunicațiile prin satelit și comunicațiile terestre este instructivă. Cu numai 20 de ani în urmă s-ar fi putut crede că viitorul va aparține comunicațiilor prin satelit. La urma urmei, sistemul telefonic s-a schimbat puțin în ultimii 100 de ani și nici nu dă semne de schimbare în următorii 100 de ani. Această evoluție lentă s-a datorat în mare măsură mediului înconjurător, în care companiilor de telefoane li se cerea să furnizeze un serviciu vocal calitativ la un preț rezonabil (ceea ce au și făcut) în schimbul unui profit garantat al investițiilor lor. Prin urmare, pentru cei care

aveau date de transmis, erau disponibile modemuri de 1200 bps. Aceasta era destul de bine pentru ceea ce exista atunci.

Introducerea competiției în 1984 în Statele Unite și, ceva mai târziu, în Europa a schimbat radical situația. Companiile de telefoane au început înlocuirea cu fibre optice a rețelelor exploatate un timp atât de îndelungat și au introdus servicii cu lățimi de bandă ridicate, cum este ADSL (Asymetric Digital Subscriber Line, rom: linie de abonat digitală asimetrica). În plus, și-au încetat practica îndelungată de a factura utilizatorii aflați la distanțe mai mari cu sume mărite artificial pentru a subvenționa utilizatorii locali.

Dintr-o dată, se părea că legăturile terestre pe fibră optică reprezintă câștigătorul pe termen lung. Cu toate acestea, sateliții de comunicație au câteva nișe pe piață, în care fibra optică nu a pătruns (și în unele cazuri nici nu o va putea face). Vom analiza acum o parte dintre acestea.

În timp ce o singură fibră optică are, în principiu, mai multă lățime potențială de bandă decât toți sateliții lansați vreodată, această lățime de bandă nu este disponibilă majorității utilizatorilor. Fibrele optice instalate la ora actuală sunt folosite în sistemul telefonic pentru a gestiona simultan mai multe apeluri de distanță lungă, și nu pentru a furniza utilizatorilor individuali lățime de bandă ridicată. În cazul sateliților, un utilizator poate foarte bine să scoată o antenă pe acoperișul clădirii și să ocolească sistemul telefonic. Teledesic se bazează pe această idee.

O a doua nișă o reprezintă comunicațiile mobile. În zilele noastre, mulți oameni doresc să comunice în timp ce fac jogging, conduc, navighează sau zboară. Legăturile terestre prin fibre optice nu le sunt de nici un folos, în schimb le pot fi utile legăturile prin satelit. Este posibil, totuși, ca o combinație între radioul celular și fibra optică să fie satisfăcătoare pentru cerințele majorității utilizatorilor (probabil cu excepția acelora care se află la bordul unui avion sau pe mare).

O a treia nișă o reprezintă situațiile în care este esențială difuzarea. Un mesaj transmis de satelit poate fi recepționat simultan de mii de stații terestre. De exemplu, o firmă care transmite acțiuni, titluri de proprietate sau prețurile mărfurilor către mii de distribuitori, ar putea descoperi că utilizarea unui sistem prin satelit este mult mai ieftină decât simularea difuzării pe Pământ.

O a patra nișă o constituie comunicația în locurile cu terenuri greu accesibile sau cu o infrastructură terestră slab dezvoltată. Indonezia, de exemplu, are propriul satelit pentru traficul telefonic intern. Lansarea unui satelit a fost mult mai simplă decât întinderea a mii de cabluri submarine între cele 13.677 de insule din arhipelag.

O a cincea nișă pentru piața sateliților este acolo unde dreptul de instalare a fibrei optice este dificil de obținut sau nejustificat de scump.

În al șaselea rând, atunci când instalarea rapidă este critică, cum este în cazul sistemelor de comunicații militare pe timp de război, sateliții obțin câștig de cauză fără probleme.

Pe scurt, se pare că în viitor fluxul principal de comunicație va fi pe fibră optică în combinație cu radioul celular, iar pentru câțiva utilizatori specializați, vor rămâne preferabili sateliții. Totuși, există un avertisment valabil pentru toate acestea: economia. Cu toate că fibra optică oferă mai multă lățime de bandă, este posibil, fără îndoială, ca în viitor comunicațiile terestre și cele prin satelit să intre într-o competiție agresivă pe baza prețului practicat. Dacă progresele tehnologice vor reduce radical costul de instalare al unui satelit (de exemplu, unele viitoare navete spațiale vor putea împrăștia în spațiu mai multe zeci de sateliți la o singură lansare) sau dacă vor deveni populari sateliții de joasă altitudine, atunci s-ar putea ca fibrele optice să-și piardă, pe unele piețe, poziția lor de lider.

2.5 SISTEMUL TELEFONIC

Două calculatoare ale aceleiași companii sau organizații, aflate la mică distanță, pot fi conectate simplu printr-un cablu, pentru a comunica între ele. Acesta este modul de funcționare al rețelelor locale. Oricum, când distanțele sunt mari sau sunt multe calculatoare, ori când cablurile ar trebui să treacă printr-un loc public, costul instalării de cabluri particulare este de obicei prohibitiv. Mai mult, în aproape toate țările din lume, instalarea de cabluri de-a lungul (sau pe sub) proprietățile publice este ilegală. În consecință, proiectanții de rețele trebuie să se bazeze pe facilitățile de comunicație existente.

O astfel de facilitate este **PSTN**, (**Public Switched Telephone Network**, rom: rețea telefonică publică comutată), care a fost proiectată cu mulți ani în urmă, în cu totul alt scop: transmisia vocii umane într-o formă mai mult sau mai puțin recognoscibilă. Acest sistem nu este destul de potrivit pentru comunicațiile între calculatoare, dar situația se schimbă rapid odată cu introducerea fibrelor optice și a tehnologiei digitale. În orice caz, sistemul telefonic este atât de strâns legat de rețelele de calculatoare (larg răspândite geografic), încât merită să îi acordăm mai multă atenție.

Pentru avea o idee despre ordinul de mărime al problemei, să facem o comparație scurtă dar semnificativă între proprietățile unei conexiuni tipice între calculatoare printr-un cablu local și printr-o linie telefonică. Un cablu care face legătura între două calculatoare poate transfera date la 10^9 bps, poate și mai mult. Prin contrast, o linie telefonică are o viteza maximă de transfer de date de 56 Kbps, o viteză de aproape 20.000 de ori mai mică. Aceasta este diferența dintre o rață care merge legănându-se în tihnă prin iarba și o rachetă care zboară spre lună. Dacă linia telefonică este înlocuită de o conexiune ADSL, viteza este mai mică de 1000-2000 de ori.

Desigur, proiectanții sistemelor de calculatoare cheltuiesc mult timp și efort pentru a analiza cum pot fi acestea folosite cât mai eficient și au dificultăți cu un sistem a cărui performanță (din punctul lor de vedere) este cu 3 sau 4 ordine de mărime mai slabă. În paragrafele care urmează vom descrie sistemele telefonice și vom prezenta istoria și viitorul lor. Pentru informații suplimentare despre structura sistemelor telefonice vezi (Bellamy 1991).

2.5.1 Structura sistemului telefonic

Curând după ce Alexander Graham Bell a brevetat telefonul în 1876 (doar cu câteva ore înaintea rivalului său, Elisha Gray), cererea pentru noua sa invenție a fost imensă. Piața inițială consta în vânzarea telefoanelor, existente numai sub formă de perechi. Era la latitudinea clientului să întindă un fir între ele. Dacă proprietarul unui telefon dorea să comunice cu alți *n* proprietari de telefoane, trebuiau folosite fire separate pentru conectarea tuturor celor *n* case. În mai puțin de un an, orașele erau acoperite cu fire care treceau peste case și copaci într-o încrengătură sălbatică. A devenit imediat evident că modelul conectării fiecărui telefon la fiecare alt telefon, ca în fig. 2-14(a), nu va putea funcționa.

Bell a observat acest lucru și a înființat Bell Telephone Company, companie care a deschis primul oficiu de comutare (în New Haven, Connecticut), în 1878. Compania a întins câte un fir către casa sau biroul fiecărui client. Pentru a da un telefon, clientul lovea furca, generând astfel un semnal sonor în centrală, care atrăgea atenția operatorului; acesta conecta apoi manual cei doi clienți cu ajutorul unui cablu. Modelul unui oficiu de comutare este ilustrat în fig. 2-20(b).

Destul de repede, oficiile de comutare Bell Systems au apărut peste tot și oamenii au simțit nevoia unor convorbiri interurbane, oficiile Bell System începând să se conecteze între ele. Problema inițială a redevenit actuală: conectarea fiecărui oficiu de comutare cu fiecare alt oficiu prin intermediul unui cablu a scăpat rapid de sub control, și astfel s-au inventat oficiile de comutare de nivelul doi. După un timp, au fost necesare mai multe oficii de nivelul doi, ca în fig. 2-20(c). În cele din urmă, ierarhia a ajuns până la 5 niveluri.

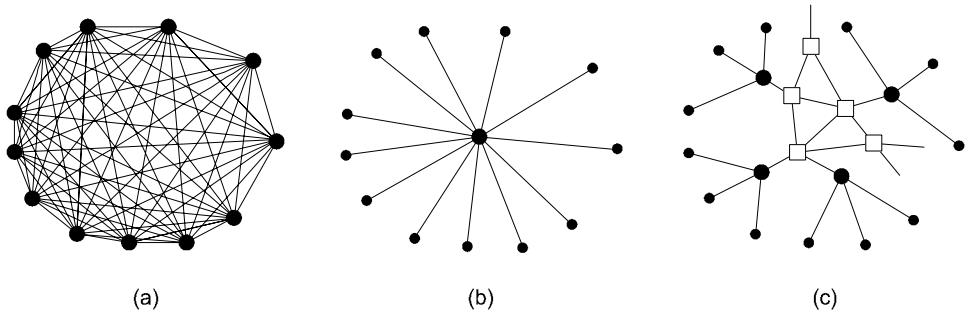


Fig. 2-20. (a) Rețea conectată integral. (b) Comutator centralizat. (c) Ierarhie pe două niveluri.

În 1890, cele trei componente majore ale sistemului telefonic erau puse la punct: oficiile de comutare, cablurile între clienți și oficiile de comutare (acum echilibrate, izolate, cablu torsadat în locul firelor neizolate, legate la pământ) și legăturile între oficiile de comutare pe distanță lungă. Cu toate că au apărut îmbunătățiri în toate cele trei domenii, modelul de bază al sistemului Bell a rămas esențialmente intact mai bine de 100 de ani. Pentru o scurtă istorie a sistemului telefonic, vezi (Hawley, 1991).

Anterior pătrunderii în forță pe piață a companiei AT&T în 1984, sistemul telefonic era implementat ca o ierarhie pe multe niveluri cu un grad mare de redundanță. Descrierea care urmează, deși foarte simplificată, conține totuși esențialul. Fiecare telefon are două fire de cupru conectate direct la cel mai apropriat **oficiu final** (deseori numit **oficiu central local**). Distanța este în mod uzual între 1 și 10 Km, fiind mai mică în orașe decât în zonele rurale. Numai în Statele Unite sunt peste 19.000 de oficii finale. Concatenarea codului zonei și a primelor trei cifre din numărul de telefon specifică în mod unic un oficiu final. Legătura formată de cele două fire între un telefon și oficiul final corespunzător este cunoscută în termeni tehnici sub numele de **buclă locală**. Dacă toate buclele locale din toată lumea ar fi fost puse cap la cap, ele ar acoperi distanța de la pământ la Lună și înapoi de 1000 de ori.

La un moment dat, 80 la sută din capitalul AT&T era constituit de cuprul din buclele locale. AT&T era atunci, de fapt, cea mai mare mină de cupru din lume. Din fericire, acest lucru nu era prea mult cunoscut în lumea investițiilor. Dacă s-ar fi cunoscut, AT&T putea fi cumpărată, lichidate toate serviciile telefonice din Statele Unite, smuls tot cablul și vândut unei rafinării de cupru pentru un profit imediat.

Dacă un abonat atașat la un anumit oficiu final apelează alt abonat atașat la același oficiu final, mecanismul de comutare din acel oficiu stabilește o legătură electrică directă între cele două bucle locale. Această legătură rămâne intactă pe toată durata convorbirii.

Dacă telefonul apelat este atașat la un alt oficiu final, trebuie folosită o altă procedură. Fiecare oficiu final are un număr de linii conectate la unul sau mai multe centre de comutare apropriate, numite **oficii de taxare** (sau, dacă sunt în aceeași zonă, **oficii în tandem**). Aceste linii se numesc **trunchiuri de conectare la oficiile de taxare** (**toll connecting trunks**). Dacă se întâmplă ca oficiul final al celui care apelează și oficiul celui apelat să aibă trunchi de conectare către același oficiu de taxare (ceea ce este probabil dacă sunt relativ apropriate), legătura poate fi stabilită de către oficiul de taxare. O rețea telefonică formată din telefoane (punctele mici), oficii finale (punctele mari) și oficiile de taxare (pătratele) este prezentată în fig. 2-20(c).

Dacă apelantul și apelatul nu au un oficiu de taxare în comun, calea va trebui să fie stabilită undeva mai sus în ierarhie. Oficiile de taxare sunt conectate prin intermediul unei rețele formate din oficii primare, sectoriale și regionale. Comunicațiile între oficiile de taxare primare, sectoriale și regionale se realizează prin intermediul **trunchiurilor de comunicație** de bandă foarte largă (numite și **trunchiuri de comunicație inter-oficii**). Varietatea centrelor de comutare și a topologiei acestora (pot două oficii sectoriale să fie conectate direct sau prin intermediul unui oficiu regional ?) diferă de la o țară la alta în funcție de densitatea telefonică. Fig. 2-21 prezintă un mod posibil de a realiza o legătură pe distanțe medii. În telecomunicații sunt folosite diverse medii de transmisie. În prezent, buclele locale constau din cabluri torsadate de categoria 3 – cu toate că în primele zile ale telefoniei erau uzuale firele neizolate – aflate la o distanța de 25 cm între ele, la polii telefonului. Între oficiile de comutare sunt larg folosite cablurile coaxiale, microundele și mai ales fibrele optice.

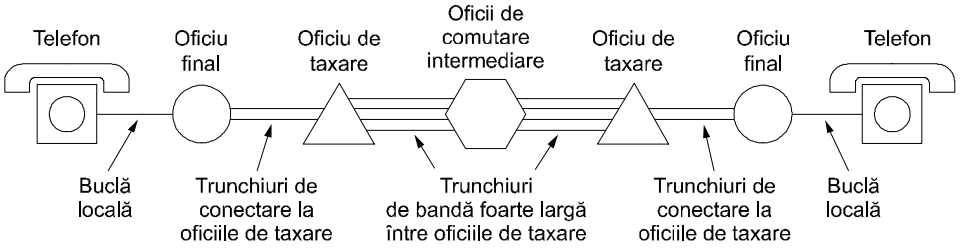


Fig. 2-21. Circuitul unei rute uzuale pentru o convorbire la distanță medie

În trecut, transmisia în sistemul telefonic era analogică, semnalul vocal fiind transmis de la sursă la destinație sub forma unei tensiuni electrice. Odată cu apariția fibrei optice, a electronicii digitale și a calculatoarelor, toate trunchiurile și comutatoarele sunt acum digitale, bucla locala rămânând ultima parte de tehnologie analogică a sistemului. Transmisia digitală este preferată, nefiind necesar să se reproducă exact o formă de undă analogică după ce a trecut prin mai multe amplificatoare într-o convorbire la distanță. Distincția între 1 și 0 este suficientă. Această proprietate face ca transmisia digitală să fie de mai mare încredere decât cea analogică. De asemenea, este mai ieftină și mai ușor de întreținut.

Pe scurt, sistemul telefonic constă din trei componente majore:

1. Bucle locale (cablu torsadat analogic tras în interiorul caselor și al companiilor).
2. Trunchiuri (fibre optice digitale care conectează oficiile de comutare).
3. Oficii de comutare (unde apelurile sunt transferate dintr-un trunchi în altul).

După o scurtă prezentare a politicii în domeniul telefonic, vom studia aceste trei componente mai în detaliu. Buclele locale oferă tuturor acces la întregul sistem, deci sunt critice. Din păcate, ele

reprezintă și veriga cea mai slabă a sistemului. Pentru trunchiurile pe distanțe mari, problema principală va fi gruparea mai multor convorbiri împreună și trimiterea lor simultană. Soluția se numește multiplexare și vom studia trei modalități diferite de multiplexare. În sfârșit, există două moduri fundamental diferite de a face comutarea, așa că le vom studia pe amândouă.

2.5.2 Politica din domeniul telefonic

Timp de mai multe zeci de ani, până în 1984, Bell Systems a asigurat servicii, atât pe distanțe scurte cât și pe distanțe lungi, pentru aproape toată suprafața Statelor Unite. În anii 70, guvernul S.U.A. a ajuns la concluzia că acesta era un monopol ilegal și a hotărât să îl anuleze. Guvernul a câștigat pe 1 ianuarie 1984, AT&T fiind destrămată în AT&T Long Lines, 23 de companii **BOC** (**Bell Operating Companies**) și alte câteva părți. Cele 23 de companii BOC erau grupate în șapte BOC regionale (RBOC) pentru a le face viabile din punct de vedere economic. Întreaga natură a telecomunicațiilor în Statele Unite a fost schimbată peste noapte de o hotărâre judecătorească (*nu* de un act al Congresului).

Detaliile exacte ale acestei privațiuni sunt descrise în așa numita **MFJ** (**Modified Final Judgement**, rom: hotărâre finală modificată, un bun exemplu de oximoron[5], dacă a existat vreodată vreunul – dacă hotărârea putea fi modificată, este evident că nu era finală). Acest eveniment a condus la o creștere a competiției, la asigurarea unor servicii mai bune și la scăderea prețurilor pentru clienți. Totuși, prețurile pentru serviciile locale au crescut și subvențiile încrucișate de la apelurile de lungă distanța au fost eliminate, iar serviciile locale au trebuit să se susțină singure. Multe alte țări iau în considerare introducerea competiției după același model.

Să clarificăm cum s-a putut atinge acest scop: Statele Unite au fost împărțite în aproape 160 de **LATA** (**Local Access and Transport Areas**, rom: zone de acces și transport local). Pe scurt, o LATA acoperă o suprafață echivalentă ca dimensiuni cu o regiune acoperită de un același cod zonal. De obicei, în cadrul unei LATA există un **LEC** (**Local Exchange Carrier**, rom: transportator local), care deține monopolul pentru un serviciu telefonic tradițional din interiorul regiunii sale. Cele mai importante LEC sunt BOC-urile, cu toate că unele LATA conțin una sau mai multe din cele peste 1500 de companii telefonice independente care funcționează ca LEC-uri.

Tot traficul inter-LATA este asigurat de un alt tip de companie, **IXC** (**IntereXchange Carrier**, rom: transportator inter-oficii). Inițial, AT&T Long Lines era singura companie IXC serioasă, dar acum WorldCom și Sprint sunt competitori consacrați în domeniu. Una dintre preocupările apărute la separarea companiilor a fost ca toate IXC să fie tratate egal în ce privește calitatea liniilor, tarifele impuse și numărul de cifre pe care un client trebuie să le formeze pentru a telefona. Modul în care acest lucru a fost îndeplinit este prezentat în fig. 2-22. Aici vedem trei exemple de LATA, fiecare cu mai multe oficii finale. LATA-urile 2 și 3 au de asemenea o mică ierarhie formată din oficii tandem (oficii intra-LATA).

Orice IXC care dorește să asigure convorbiri provenite dintr-o LATA poate construi un oficiu de comutare, numit **POP** (**Point of Presence**, rom: punct de livrare). LEC-ul trebuie să conecteze fiecare IXC la fiecare oficiu final, direct, ca în LATA 1 și 3, sau indirect, ca în LATA 2. Mai mult, condițiile în care se face această conectare, atât tehnice cât și financiare, trebuie să fie aceleași pentru toate IXC-urile. În acest mod, un abonat din LATA 1, să zicem, poate alege ce IXC să folosească pentru a apela abonați din LATA 3.

---

[5] Oximoron = contradicție evidentă între termenii expresiei (n.t.)

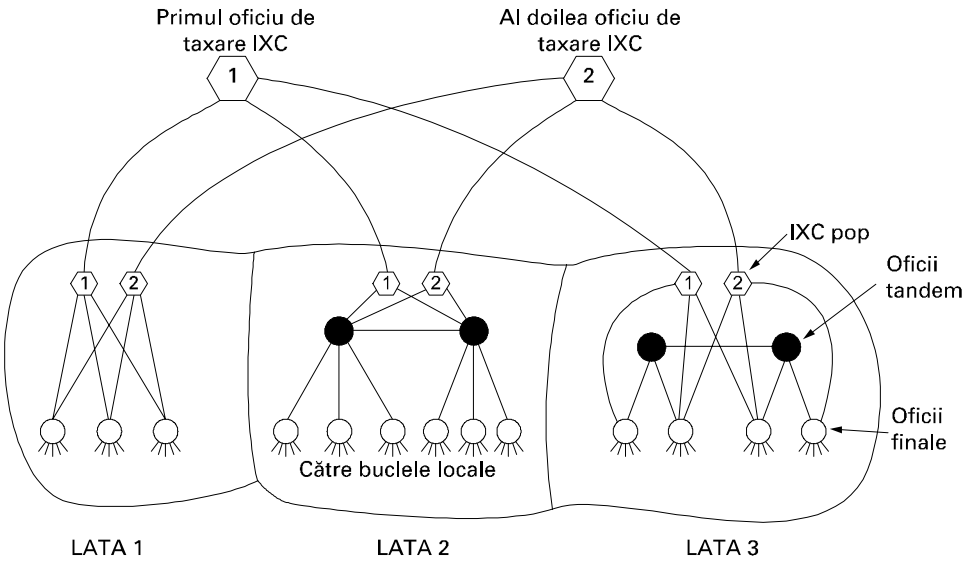


Fig. 2-22. Relația dintre LATA-uri, LEC-uri și IXC-uri. Toate cercurile sunt oficii de comutare LEC. Fiecare hexagon aparține câte unui IXC.

Printr-o clauză a MFJ, companiilor IXC le era interzis să ofere servicii telefonice locale și tuturor LEC le era interzis să ofere servicii telefonice inter-LATA, cu toate că amândouă erau libere să intre în orice altă afacere, cum ar fi chiar deschiderea de restaurante. În 1984, această declarație era lipsită de ambiguități. Din păcate, tehnologia are un fel de a face ca legea să pară depășită. Nici televiziunea prin cablu și nici telefoanele celulare nu erau acoperite de această înțelegere. Atunci când televiziunea prin cablu a trecut de la forma unidirecțională la cea bidirecțională și când a crescut brusc popularitatea telefoanelor celulare, atât LEC-urile cât și IXC-urile au început să cumpere sau să se asocieze cu companiile din aceste domenii.

În 1995, Congresul a observat că menținerea unei deosebiri între diferite tipuri de companii nu mai era posibilă și a aprobat o notă prin care permitea companiilor de cablu TV, companiilor telefonice locale, transportatorilor pe distanțe mari și operatorilor celulari să facă afaceri unul în domeniul celuilalt. Ideea era ca orice companie să poată oferi clienților săi un serviciu complet, cuprinzând cablul TV, telefon și servicii informaționale și că diferite companii ar putea concura în ceea ce privește calitatea serviciilor asigurate și prețul acestora. Această notă a fost legiferată în februarie 1996. În urma acestei hotărâri, unele BOC au devenit IXC și alte companii, precum operatorii de televiziune prin cablu, au început să ofere servicii telefonice locale în competiție cu LEC-urile.

O proprietate interesantă a legii din 1996 este cerința ca LEC-urile să implementeze portabilitatea locală a numerelor. Aceasta înseamnă ca un client poate schimba companiile locale de telefoane fără să trebuiască să obțină un nou număr de telefon. Aceasta prevedere scutește mulți clienți de o problemă serioasă și îi poate ajuta în decizia de a schimba LEC-ul, ceea ce conduce la creșterea competiției. Ca rezultat, peisajul telecomunicațiilor din SUA trece printr-o restructurare radicală. Din nou, multe alte țări încep să urmeze acest model. Deseori, celelalte țări așteaptă să vadă cum funcționează un experiment în SUA. Dacă funcționează bine, fac același lucru; dacă funcționează rău, încearcă altceva.

2.5.3 Bucla locală: Modemuri, ADSL și transmisia fără fir

Este timpul să pornim studiul nostru detaliat despre funcționarea sistemului telefonic. Principalele părți ale sistemului sunt ilustrate în figura2-23. Aici se văd buclele locale, trunchiurile, oficiile de taxare și oficiile finale, ambele conținând echipamente de comutare care comută apelurile. Un oficiu final are până la 10.000 de bucle locale (în SUA și alte țări mari). De fapt, până de curând, codul și prefixul de zonă indicau oficiul final, astfel încât numărul (212) 601-xxxx aparținea unui anumit oficiu final cu 10.000 de abonați, numerotați de la 0000 la 9999. Odată cu evoluția competiției pentru serviciile locale, acest sistem nu mai era viabil, deoarece mai multe companii doreau să aibă codul oficiului final. De asemenea, numărul de coduri era practic epuizat, astfel încât au trebuit introduse scheme de organizare mai complexe.

Să începem cu partea cu care cei mai mulți oameni sunt familiarizați: bucla locală, formată din două fire care vin de la un oficiu final al unei companii telefonice și intră în case sau companii mai mici. Bucla locală mai este numită adesea și „ultima milă", deși lungimea ei poate fi de până la câteva mile. În ultimii 100 de ani, bucla locală a folosit semnalizarea analogică și probabil va continua să o folosească timp de ani buni, datorită costului mare al conversiei la digital. Totuși, până și în acest ultim bastion al transmisiei analogice au loc schimbări. În acest capitol, vom studia bucla locală tradițională și noile îmbunătățiri care au loc în acest domeniu, concentrându-ne pe comunicația de date de la calculatoarele casnice.

Atunci când un calculator dorește să trimită date numerice pe o linie telefonică, datele trebuie să fie convertite în prealabil în formă analogică pentru a putea fi transmise pe o buclă locală . Aceasta conversie este făcută de către un modem, dispozitiv pe care îl vom studia în curând. La oficiul final al companiei telefonice, aceste date sunt convertite la forma digitală pentru a fi transmise pe trunchiurile pentru distanțe mari.

Dacă la celălalt capăt se află un calculator cu un modem, este necesara conversia inversă – digital la analogic pentru a putea traversa bucla locală către destinație. Această schemă este prezentată în fig. 2-23 pentru ISP-ul 1 (Internet Service Provider, rom: furnizor de servicii internet), care dispune de o bancă de modem-uri, fiecare fiind conectat la o altă buclă locală. Acest ISP poate servi atâtea conexiuni câte modemuri are (presupunând că serverul sau serverele sale au destulă putere de calcul). Această schemă era considerată normală până când au apărut modem-urile de 56 Kbps, din motive care se vor vedea în curând.

Codificarea analogică a semnalului constă în modificarea tensiunii electrice în funcție de timp, pentru de a reprezenta un șir de date. Dacă mediul de transmisie ar fi fost ideal, receptorul ar fi primit exact același semnal pe care l-a expediat transmițătorul. Din păcate, mediile nu sunt perfecte, iar semnalul recepționat nu este identic cu semnalul transmis. Pentru datele numerice, aceste diferențe pot conduce la erori.

Pe liniile de transmisie apar trei mari probleme: atenuarea, distorsiunea datorată întârzierii și zgomotul. **Atenuarea** reprezintă pierderea de energie în timpul propagării semnalului. Pierderea se exprimă în decibeli pe kilometru. Energia pierdută depinde de frecvența semnalului. Pentru a vizualiza efectul acestei dependențe de frecvență, să ne imaginăm un semnal nu ca o simplă undă, ci sub forma unei serii de componente Fourier. Fiecare componentă este atenuată diferit, ceea ce are ca rezultat la receptor un spectru Fourier diferit.

Pentru a agrava situația, diferitele componente Fourier se propagă cu viteze diferite de-a lungul firului. Aceste diferențe de viteză duc la **distorsionarea** semnalului recepționat la celălalt capăt.

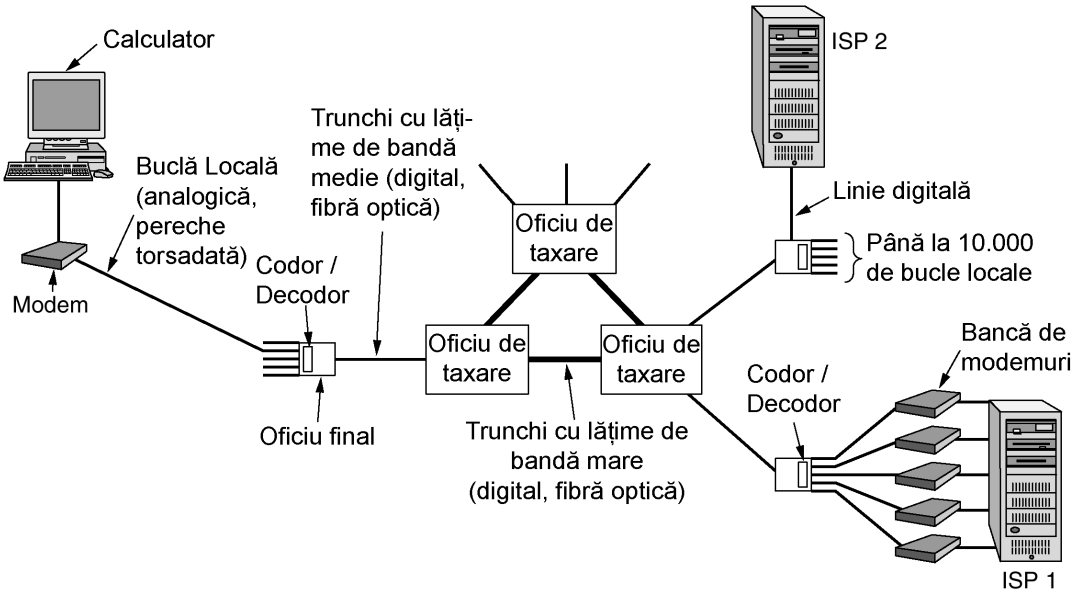


Fig. 2-23. Conectarea între calculatoare folosește transmisia analogică și cea digitală. Conversia este realizată de către modemuri și de către codoare/decodoare.

O altă problemă este **zgomotul**, care reprezintă energie nedorită, provenită din alte surse decât transmițătorul. Zgomotul termic este cauzat de mișcarea aleatorie a electronilor printr-o sârmă și nu se poate evita. Interferența este produsă de cuplajul inductiv care se formează între două fire care sunt apropriate unul de altul. Atunci când vorbim la telefon, putem auzi o altă conversație în fundal. Aceasta este interferența. În sfârșit, există și zgomote de tip impuls, determinate de șocuri electrice sau de alte cauze. Pentru datele digitale, zgomotele de tip impuls pot duce la dispariția unuia sau a mai multor biți.

Modemurile

Datorită problemelor prezentate anterior, în special datorită faptului că atât atenuarea cât și viteza de propagare sunt dependente de frecvență, se dorește evitarea prezenței unui domeniu larg de frecvențe într-un semnal.

Din păcate, undele pătratice, precum cele din datele numerice, au un spectru larg și, în concluzie, suferă o atenuare puternică și distorsiuni de întârziere. Aceste efecte fac din codificarea analogică în bandă de bază (DC) o alegere nepotrivită, cu excepția situațiilor în care se utilizează viteze mici și transmisia are loc pe distanțe scurte.

Pentru a evita problemele asociate cu codificarea analogică în bandă de bază (DC), în special pe liniile telefonice, se utilizează codificarea analogică AC. Se introduce un ton continuu în domeniul 1000 - 2000 de Hz, numit **undă purtătoare sinusoidală**. Amplitudinea, frecvența sau faza acestei unde pot fi modulate. În **modularea în amplitudine**, sunt folosite două niveluri de tensiune pentru a reprezenta 0 și 1, respectiv. În **modularea în frecvență**, cunoscută de asemenea sub denumirea de **codare prin deplasarea frecvenței** (**frequency shift keying**), se folosesc două (sau mai multe) tonuri diferite. (Termenul de **codare** este larg folosit în industrie ca sinonim pentru modulare). În varianta cea mai simplă, cea a **modulării în fază**, unda purtătoare este sistematic comutată la intervale egale

la 45, 135, 225, sau 315 grade. Fiecare schimbare de fază transmite 2 biți de informație. De asemenea, obligativitatea unei schimbări de fază la sfârșitul fiecărui interval face receptorul să recunoască mai ușor limitele intervalelor de timp.

Fig. 2-24 ilustrează cele trei forme de modulare. În figura2-24(a) una dintre amplitudini este diferită de zero și una este zero. În fig. 2-24(b) sunt folosite două frecvențe. În fig. 2-24(c) o deplasare de fază este sau nu prezentă la marginile fiecărui bit. Un echipament care acceptă un șir serial de biți la intrare și produce o purtătoare modulată la ieșire (sau vice-versa) se numește **modem** (modulator-demodulator). Modemul este inserat între calculator (digital) și sistemul telefonic (analogic).

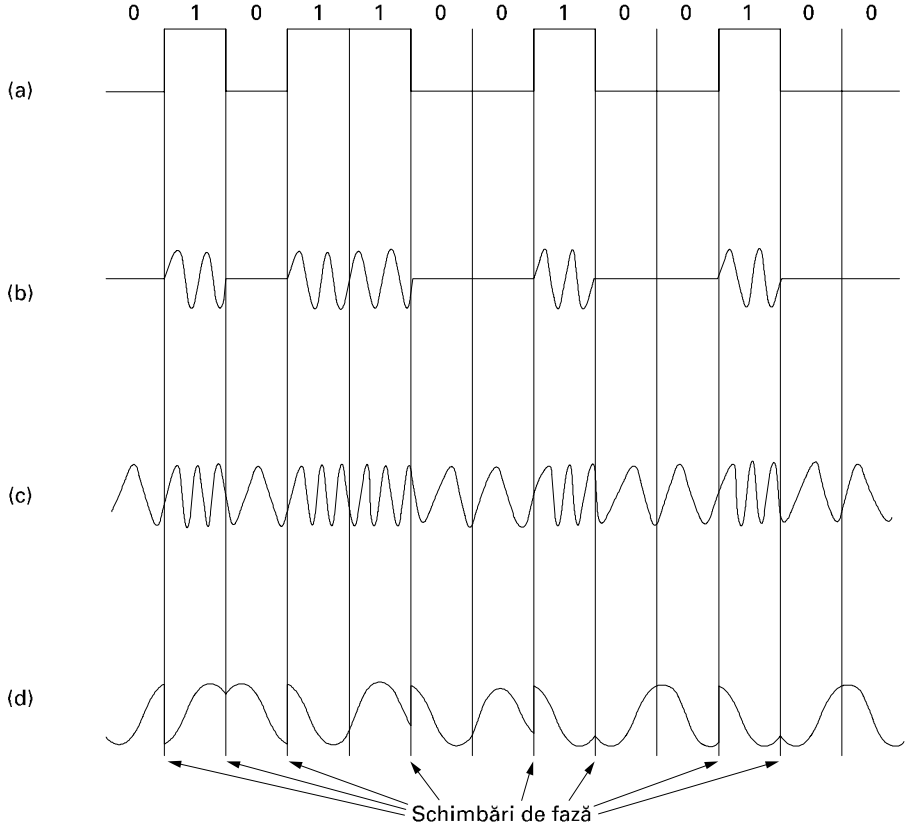


Fig. 2-24. (a) Un semnal binar. (b) Modularea în amplitudine. (c) Modularea în frecvență. (d) Modularea în fază.

Atingerea unor viteze din ce în ce mai mari nu este posibilă doar prin creșterea continuă a ratei de eșantionare. Teorema lui Nyquist afirmă că eșantionarea la o frecvența mai mare de 6000 de Hz este lipsită de interes chiar și pentru o linie ideală de 3000 de Hz (nu este nici pe departe cazul unei linii telefonice). În practică, majoritatea modem-urilor eșantionează de 2400 ori/sec și se concentrează să transmită cât mai mulți biți pe eșantion.

Numărul de eșantioane pe secundă se măsoară în **baud**. Pe durata fiecărui baud este trimis un simbol. Astfel, o linie de *n* baud trimite *n* simboluri/sec. De exemplu, o linie de 2400 baud trimite un simbol la aproximativ fiecare 416,667μsec. Dacă simbolul codifică prin tensiune nulă un bit 0 logic și

prin tensiunea de 1V un bit 1 logic, rata de bit este de 2400 bps. Dacă însă sunt folosite tensiunile de 0, 1, 2 și 3 volți, fiecare simbol codifică 2 biți, astfel că o linie de 2400 baud poate transmite 2400 simboluri/sec la o rată de date de 4800 bps. În mod similar, pentru patru deplasări de fază posibile, sunt codificați tot 2 biți/simbol, deci avem din nou o rată de bit dublă față de viteza de transmisie a liniei. Cea de-a doua tehnică este larg folosită și se numește **QPSK** (**Quadrature Phase Shift Keying**, rom: modulația cuadratică în fază).

Conceptele de lărgime de bandă, viteza de transmisie (eng: baudrate), rată de simboluri și rată de biți sunt adeseori confundate, deci le vom reformula aici. Lărgimea de bandă a unui mediu reprezintă spectrul de frecvențe care trec prin el cu atenuare minima. Este o proprietate fizică a mediului (de obicei de la 0 la o frecvență maximă) și se măsoară în Hz. Viteza de transmisie reprezintă numărul de eșantioane preluate într-o secundă. Prin fiecare eșantion se transmite o parte din informație, adică un simbol. Deci, viteza de transmisie și rata de simboluri sunt unul și același lucru. Tehnica de modulare (de exemplu QPSK) determină numărul de biți/simbol. Rata de biți reprezintă cantitatea de informație trimisă prin canal și este egală cu numărul de simboluri/sec înmulțit cu numărul de biți/simbol.

Toate modemurile performante folosesc o combinație de tehnici de modulare pentru a transmite mai mulți biți pe baud. Deseori mai multe amplitudini și mai multe deplasări de faza sunt combinate pentru a transmite mai mulți biți/simbol. În fig. 2-25(a), vedem puncte la 45, 135, 225 și 315 grade, cu amplitudine constanta (distanța față de origine). Faza unui punct este indicata de unghiul pe care l-ar face axa x cu o linie care unește punctul cu originea. Fig. 2-25(a) are patru combinații posibile și poate fi folosită pentru a transmite 2 biți pe simbol. Este exact QPSK.

În fig. 2-25(b) vedem o schemă diferită de modulare, în care sunt folosite 4 amplitudini și 4 diferențe de fază, în total 16 combinații. Această schemă de modulare poate fi folosită la transmiterea a 4 biți pe simbol și este numită **QAM** (**Quadrature Amplitude Modulation**, rom: modulația cuadratică în amplitudine) atunci când este folosită pentru transmisia a 9600 biți pe secundă pe o linie de 2400 baud.

Fig. 2-25(c) reprezintă încă o schemă de modulare, care implică amplitudine și faza. Ea permite 64 de combinații, astfel încât pot fi transmiși 6 biți pe simbol. Se numește **QAM-64**. Sunt folosite și **QAM**-uri de ordine mai înalte.

Diagramele de genul celor din fig. 2-25, care reprezintă combinațiile posibile de amplitudine și fază, sunt numite **tipare de constelații**. Fiecare standard de modem de viteze înalte are propriul lui tipar de constelație și poate comunica numai cu alte modemuri care folosesc același standard (cu toate că majoritatea modemurilor pot simula modemuri mai lente).

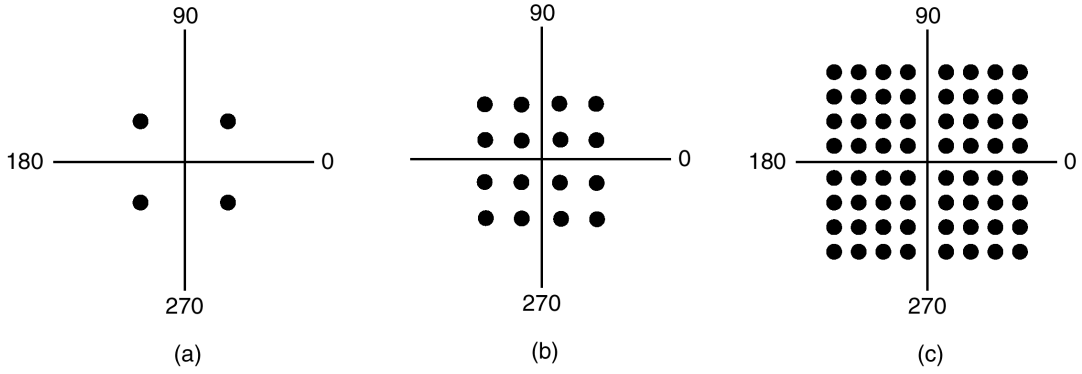


Fig. 2-25. (a) QPSK. (b) QAM-16. (c) QAM-64

Cu atâtea puncte în tiparul de constelație, chiar și un mic nivel de zgomot detectat în amplitudine sau fază poate conduce la o eroare, adică la mai mulți biți eronați. Pentru a reduce posibilitatea de a genera o eroare, standardele pentru modemuri cu viteze mari fac corecția erorilor, adăugând biți suplimentari la fiecare eșantion. Astfel de scheme sunt cunoscute sub numele de **TCM** (**Trellis Coding Modulation**, rom: modulatie prin codificare matricială). De exemplu, modemul standard V.32 folosește 32 de puncte în constelație pentru a transmite 4 biți de date și un bit de paritate pe simbol, la 2400 baud, și obține 9600 bps cu corecție de erori. Tiparul său de constelație este cel din fig. 2-26(a). Decizia de a fi „rotită" cu 45 de grade în jurul originii a fost luată din motive inginerești; constelațiile rotite sau nerotite au aceeași capacitate de informație.

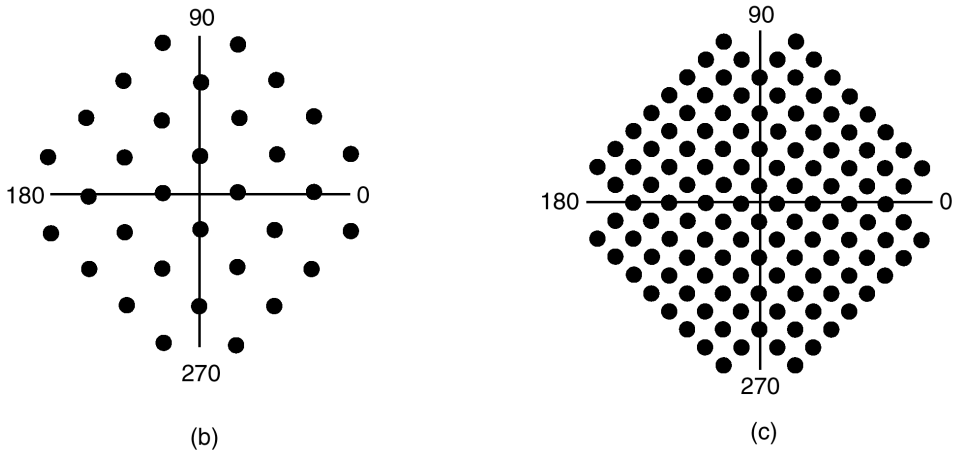


Fig. 2-26. (a) V.32 pentru 9600 Kbps. (b) V.32 bis pentru 14.400 Kbps

Următorul pas peste 9600 bps este 14.400 bps. Este numit **V.32 bis**. Această viteză este atinsă prin transmiterea a 6 biți de date și un bit de paritate pe eșantion la o rată de 2400 baud. Tiparul de constelație are 128 de puncte atunci când se folosește **QAM-128**, așa cum se vede în fig. 2-26(b). Fax-modemurile folosesc această viteză pentru a transmite pagini care au fost scanate ca o hartă de biți (bitmap). QAM-256 nu este folosit în nici unul din modemurile telefonice standard, dar este folosit în rețelele de cablu, așa cum vom vedea mai târziu.

Următorul modem telefonic după V.32 bis este **V.34**, care atinge 28.800 bps la 2400 baud cu 12 biți de date/simbol. Ultimul modem în aceasta serie este **V.34 bis**, care folosește 14 biți de date/simbol la 2400 baud și atinge 33.600 baud.

Pentru a crește în continuare rata efectivă, multe modemuri comprimă datele înainte de a le transmite, și pot obține o rată efectivă de transmisie a datelor de peste 33.600 bps. Pe de alta parte, aproape toate modemurile testează linia înainte de a începe să transmită date de la utilizator și, dacă observă că sunt probleme care țin de calitatea transmisiei, reduc viteza sub cea maxima. Astfel, viteza efectivă observată de utilizator poate fi mai mică, egală sau mai mare față de cea oficială.

Toate modemurile moderne permit traficul din ambele direcții în același timp (folosind frecvențe diferite pentru direcții diferite). O conexiune care permite traficul simultan în ambele direcții se numește **full duplex**. O șosea cu două benzi este full duplex. O conexiune care permite traficul în oricare dintre sensuri, dar pe rând se numește **half duplex**. O șină de cale ferata este **half duplex**. O conexiune care permite traficul într-o singura direcție se numește **simplex**. O stradă cu sens unic este

simplex. Un alt exemplu de conexiune simplex este o fibră optică cu un laser la un capăt și un detector de lumină la celălalt.

Motivul pentru care modemurile standard se opresc la 33.600 este că limita Shannon pentru sistemul telefonic este de aproximativ 35 Kbps, așa că o viteza mai mare ar încălca legile fizicii (departamentul termodinamică). Pentru a afla dacă modemurile de 56 Kbps sunt posibile teoretic, urmăriți discuția în continuare.

De ce este limita teoretică de 35 Kbps? Are de-a face cu lungimea medie a buclelor locale și cu calitatea acestor linii. Cei 35 Kbps sunt determinați de lungimea medie a buclelor locale. În fig. 2-23, un apel pornit de la computerul din stânga și terminat la ISP1 trece prin două bucle locale ca semnal analogic, una la sursă și una la destinație. Fiecare dintre acestea adaugă semnalului zgomote. Dacă am putea scăpa de una dintre aceste bucle locale, rata maximă s-ar dubla.

ISP2 face exact acest lucru. Are o conexiune pur digitală de la cel mai apropiat oficiu final. Semnalul digital folosit în trunchiuri ajunge direct la ISP 2, eliminând codoarele/decodoarele, modemurile și transmisia analogică finală. Astfel, când unul din capetele conexiunii este total digital, cum se întâmplă cu majoritatea ISP-urilor în zilele noastre, rata maxima de date poate fi până la 70 Kbps. Între doi utilizatori cu modemuri și linii analogice, rata maximă este de 33,6 Kbps.

Motivul pentru care se folosesc modemurile de 56 Kbps are de-a face cu teorema lui Nyquist. Canalul telefonic este lat de aproximativ 4000 Hz (incluzând benzile suplimentare). Numărul maxim de eșantioane independente care se pot prelua pe secunda este astfel de 8000. Numărul de biți per eșantion în SUA este 8, dintre care unul este folosit pentru comandă, permițând astfel numai 56.000 bps pentru date de la utilizator. În Europa, toți cei 8 biți sunt disponibili pentru date de la utilizator, deci puteau fi folosite modemuri de 64.000 bps, dar pentru a se conveni asupra unui standard internațional, s-a ales 56.000.

Acest standard de modem este numit **V.90**. Oferă un canal ascendent de 33,6 Kbps (utilizator către ISP) și un canal descendent de 56 Kbps (ISP către utilizator), pentru că de obicei sunt mai multe date de transportat de la ISP la utilizator decât invers (de exemplu, cererea unei pagini web ocupă doar câțiva octeți, în timp ce pagina efectivă poate fi de câțiva megabiți). În teorie, un canal ascendent de peste 33,6 Kbps era posibil, dar deoarece multe bucle locale sunt prea zgomotoase chiar și pentru 33,6 Kbps, s-a decis să se aloce mai multa lărgime de banda pentru canalul descendent, pentru a crește șansele ca acesta să ajungă să funcționeze la 56 Kbps.

Următorul pas dincolo de V.90 este **V.92**. Aceste modemuri sunt capabile să transfere cu 48 Kbps pe canalul ascendent, dacă linia telefonică suportă. De asemenea, ele determină viteza potrivită de funcționare în aproximativ jumătate din timpul uzual de 30 de secunde cerut de vechile modemuri. În sfârșit, permit unui apel telefonic să întrerupă o sesiune Internet, dacă linia telefonică suporta serviciul de apel în așteptare.

Linii digitale pentru abonat (xDSL)

Când industria telefonică a ajuns în sfârșit la 56 Kbps, s-a bătut singură pe spate ca pentru o treabă bine făcută. Intre timp, industria televiziunii prin cablu oferea viteze de până la 10 Mbps pe cabluri partajate, iar companiile de servicii prin satelit plănuiau să ofere trafic ascendent spre satelit de 50 Mbps. Cum accesul la Internet devenise o parte din ce în ce mai importantă din afacerile lor, companiile telefonice (LEC-urile) au început să înțeleagă că aveau nevoie de un produs mai competitiv. Răspunsul găsit a fost să înceapă să ofere noi servicii digitale peste bucla locală. Serviciile cu mai multa lățime de bandă decât serviciul telefonic standard sunt numite uneori **broadband** (rom: de bandă larga), deși termenul este mai mult un concept de marketing decât un real concept tehnic.

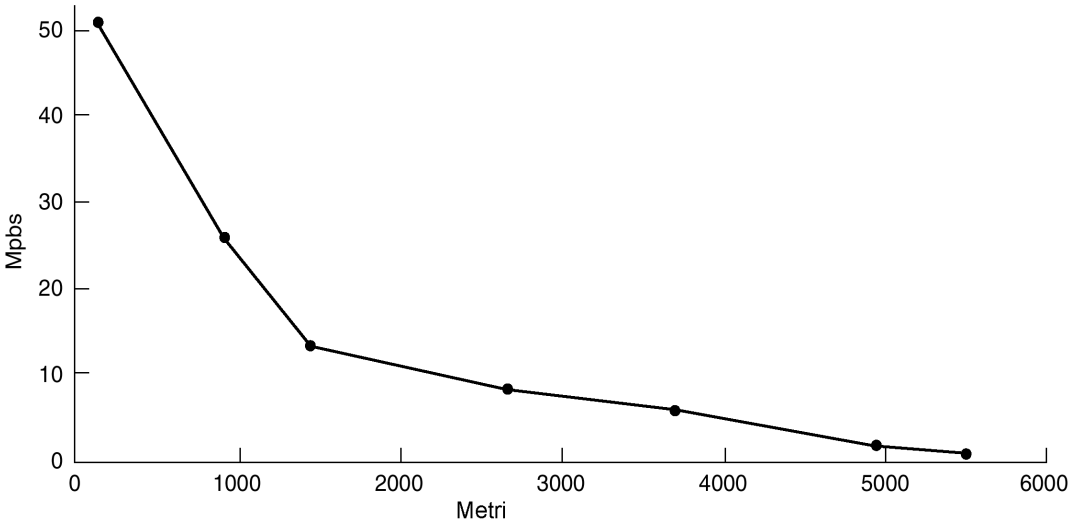


Fig. 2-27. Lățime în funcție de distanță pentru UTP categoria 3 pentru DSL.

Inițial, erau mai multe oferte care se suprapuneau, toate purtând numele generic de **xDSL** (**Digital Subscriber Line**, rom: linie digitală pentru abonat), cu diferite variabile *x*. În continuare vom discuta aceste servicii, dar ne vom concentra în primul rând spre ceea ce probabil va deveni cel mai popular dintre aceste servicii, **ADSL** (**Asymmetric DSL**, rom: DSL asimetrice). Cum ADSL încă se mai dezvolta și nu toate standardele sunt cu totul finalizate, unele dintre detaliile date mai jos s-ar putea să se schimbe în timp. Dar ideea de baza ar trebui să rămână valabilă. Pentru mai multe informații despre ADSL, se poate vedea (Summers, 1999; și Vetter et al., 2000).

Motivul pentru care modemurile sunt încete este acela că telefoanele au fost inventate pentru a transporta vocea umană și întregul sistem a fost optimizat pentru acest scop. Datele au fost mereu copiii vitregi. În punctul în care fiecare buclă locală se conectează în oficiul final, firul intra într-un filtru care atenuează toate frecvențele de sub 300 Hz și peste 3400 Hz. Filtrarea nu este exactă – 300 Hz și 3400 Hz sunt punctele de la 3 dB – așa că lărgimea de bandă este considerată de 4000 Hz, chiar dacă distanța dintre punctele de 3 dB este de 3100 Hz. Deci și datele sunt restricționate în aceasta bandă îngustă.

Trucul care face ca xDSL să funcționeze este că linia unui client abonat la un astfel de serviciu este conectată la un tip diferit de comutator, care nu are acest filtru, făcând astfel disponibilă întreaga capacitate a buclei. Factorul limitator este constitut de legile fizice aplicate buclei locale și nu de lățimea de bandă artificială de 3100 Hz, creată de filtru.

Din păcate, capacitatea buclei locale depinde de câțiva factori, inclusiv lungimea, grosimea și calitatea la modul general. Un grafic al lățimii de bandă potențiale în funcție de distanță este prezentat în fig. 2-27. Aceasta figură presupune că toți ceilalți factori sunt optimi (fire noi, legături bune, etc.).

Implicațiile acestei figuri creează o problemă pentru compania de telefoane. Când alege viteza pe care o va oferi abonaților, alege în același timp o rază de acțiune în funcție de oficiile sale finale, dincolo de care serviciul nu poate fi oferit. Aceasta înseamnă că atunci când utilizatori aflați la distanță încearcă să se înscrie la acest serviciu, s-ar putea să li se spună „Mulțumim mult pentru interesul dumneavoastră, dar locuiți cu 100 de m prea departe de cel mai apropiat oficiu final prin care puteți a beneficia de acest serviciu. Puteți să vă mutați?”. Cu cât este mai mică viteza aleasă, cu atât

mai mare este raza, fiind astfel acoperiți mai mulți clienți. Dar cu cât este mai mică viteza, cu atât este mai puțin atractiv serviciul și oamenii care vor dori să plătească pentru el vor fi mai puțini. Aici afacerile se întâlnesc cu tehnologia. (O potențială soluție este să se construiască mini-oficii finale în toate cartierele, dar aceasta este o propunere cam scumpă.)

Serviciile xDSL au fost proiectate cu anumite scopuri. În primul rând, serviciile trebuie să funcționeze peste buclele locale de cabluri cu perechi torsadate de categoria 3 existente. În al doilea rând, nu trebuie să afecteze telefoanele și faxurile clienților. În al treilea rând, trebuie să fie mult mai rapide decât 56 Kbps. În al patrulea rând, ar trebui să funcționeze tot timpul, contra unei taxe lunare, dar nu a unei taxe pe minut.

Oferta ADSL inițială a venit de la AT&T și funcționa prin divizarea spectrului disponibil în bucla locală, care este de aproximativ 1.1 MHz, în trei benzi de frecvență: **POTS** (**Plain Old Telephone Service**, rom: serviciul telefonic tradițional), canalul ascendent (de la utilizator la oficiul final) și canalul descendent (de la oficiul final la utilizator). Tehnica de a avea mai multe benzi de frecvență se numește multiplexare prin divizarea frecvenței; o vom studia în detaliu într-un paragraf ulterior. Ofertele care au urmat, de la alți furnizori, au urmat o alta abordare și se pare că aceasta va avea câștig de cauză, așa că o vom descrie în continuare.

Abordarea alternativă, numita **DMT** (**Discrete MultiTone**, rom: ton multiplu discret), este ilustrată în figura2-28. De fapt, spectrul disponibil de 1.1 MHz al buclei locale se divizează în 256 canale independente de 4312,5 Hz fiecare. Canalul 0 este folosit pentru POTS. Canalele 1-5 sunt nefolosite, pentru a preveni interferențele între semnalele de voce și date. Dintre cele 250 de canale rămase, unul este folosit pentru controlul fluxului ascendent și unul pentru controlul fluxului descendent. Restul sunt disponibile pentru date de la utilizator.

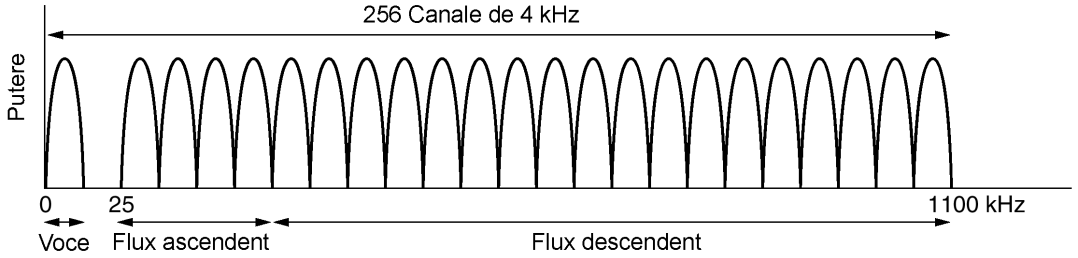


Fig. 2-28. Funcționarea ADSL folosind modulație cu ton multiplu discret

În principiu, fiecare dintre canalele rămase poate fi folosit pentru flux de date full-duplex, dar armonicele, interferențele și alte efecte țin sistemele practice departe de limitele teoretice. Este la latitudinea furnizorului de servicii să determine câte canale să fie folosite pentru fluxul ascendent și câte pentru cel descendent. Un raport de egalitate între fluxurile ascendent și descendent este tehnic posibil, dar majoritatea furnizorilor de servicii alocă în jur de 80%-90% din lățimea de bandă canalului descendent, deoarece majoritatea utilizatorilor primesc mai multe date decât trimit. Această alegere dă naștere „A”-ului din ADSL. O variantă de divizare des întâlnită este alocarea a 32 de canale pentru fluxul ascendent, restul fiind alocate pentru cel descendent. De asemenea, este posibil să se folosească unele dintre canalele cu frecvența cea mai ridicată din fluxul ascendent în mod bidirecțional, pentru creșterea lățimii de bandă, deși această optimizare cere adăugarea unui circuit special pentru anularea ecourilor.

Standardul ADSL (ANSI T1.413 și ITU G.992.1) permite viteze de până la 8 Mbps pentru fluxul descendent și 1 Mbps pentru cel ascendent. Totuși, puțini furnizori de servicii oferă acest flux. De

obicei, furnizorii oferă 512 Kbps pentru fluxul descendent și 64 Kbps pentru cel ascendent în cazul serviciului standard, respectiv 1 Mbps pentru flux descendent și 256 Kbps pentru flux ascendent în cazul serviciului premium.

În cadrul fiecărui canal este folosită o schemă de modulare similară cu V.34, deși rata de eșantionare este de 4000 baud în loc de 2400 baud. Calitatea liniilor din fiecare canal este constant monitorizata și rata de transmisie este constant ajustată, așa că pe canale diferite pot fi folosite rate diferite. Datele efective sunt trimise cu modulație QAM, cu până la 15 biți per baud, folosind o diagramă constelație analoagă cu aceea din figura2-25(b). Având, de exemplu, 224 de canale de flux descendent și 15 biți pe baud la 4000 baud, lățimea de bandă pe flux descendent este de 13,44 Mbps. În practică, raportul semnal-zgomot nu este destul de bun pentru a se atinge aceasta rată, dar se poate atinge rata de 8 Mbps pe distanțe scurte și pe bucle de calitate superioara, motiv pentru care standardul merge până la aceasta valoare.

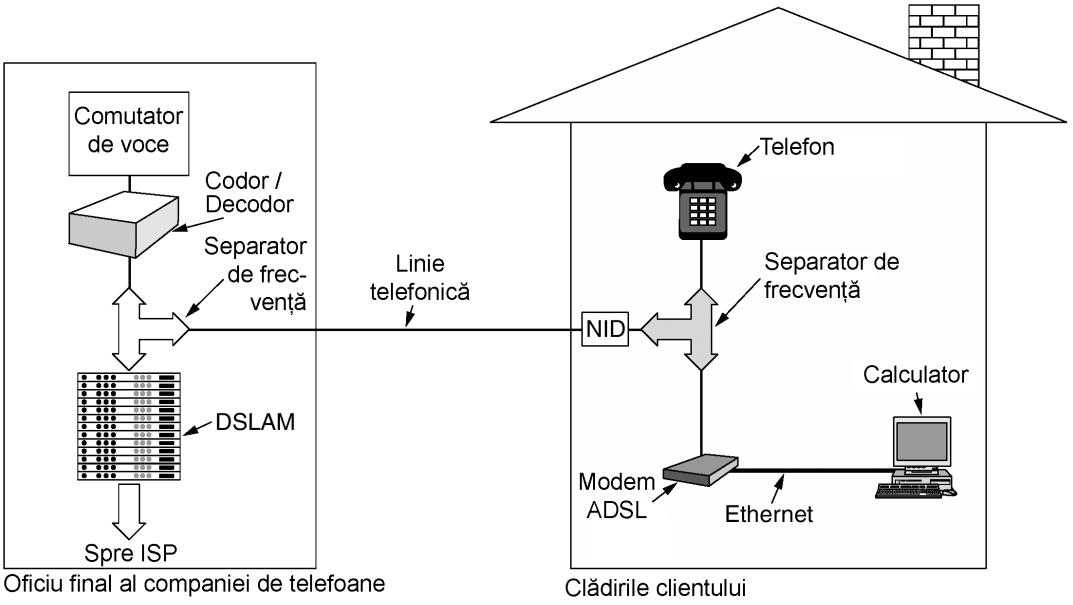


Fig. 2-29. O configurație tipică pentru echipamente ADSL.

O schema tipică ADSL este ilustrata în fig. 2-29. În aceasta schemă, un tehnician al companiei de telefoane trebuie să instaleze un **NID** (**Network Interface Device**, rom: dispozitiv de interfațare rețea) la cererea utilizatorului. Această cutie mică de plastic marchează sfârșitul proprietății companiei de telefoane și începutul proprietății clientului. Aproape de NID (sau uneori combinat cu acesta) se află un **separator** (eng. splitter), un filtru analogic care separă din fluxul de date banda de 0-4000 Hz, folosită de POTS. Semnalul POTS este dirijat la telefonul sau faxul existent, iar semnalul de date este dirijat către modemul ADSL. Modemul ADSL este de fapt un procesor de semnal digital configurat să funcționeze ca 250 de modemuri QAM care operează în paralel la diferite frecvențe. Cum majoritatea modemurilor ADSL sunt externe, calculatorul trebuie sa fie conectat cu ele la o viteza mare. De obicei, aceasta se face punând o placă Ethernet în calculator și operând o rețea Ethernet foarte mică, de două noduri, conținând doar calculatorul și modemul ADSL.

Uneori este folosit portul USB în loc de Ethernet. Pe viitor vor fi disponibile, fără îndoială, plăci interne de modem ADSL.

La celălalt capăt al firului, pe partea oficiului final, este instalat un separator de frecvențe similar. Aici este filtrată partea de voce a semnalului și este trimisă către un comutator de voce normal.

Semnalul de peste 26 KHz este rutat către un nou tip de dispozitiv numit **DSLAM** (**Digital Subscriber Line Access Multiplexer**, rom: multiplexor pentru acces la linie digitală pentru abonat), care conține același tip de procesor de semnal digital ca și modemul ADSL. Odată ce semnalul digital recepționat a fost convertit într-un șir de biți, sunt formate pachete și acestea sunt trimise ISP-ului.

Această separare completă între sistemul de voce și ADSL simplifică furnizarea serviciilor ADSL de către companiile telefonice. Nu este nevoie decât de cumpărarea unui DSLAM și a unui separator, urmată de atașarea abonaților ADSL la separator. Alte servicii de lățime mare de bandă (de exemplu ISDN) necesită schimbări mult mai mari la nivelul echipamentelor de comutare deja existente.

Un dezavantaj al soluției din figura2-29 este prezența NID-ului și a separatorului în locuința clientului. Instalarea acestora se poate face doar de către un tehnician al companiei de telefonie, fiind nevoie de o intervenție specială scumpă (adică de trimiterea unui tehnician la locuința clientului). Prin urmare, s-a standardizat o variantă alternativă fără separator. Neoficial este numit G.lite dar numărul de standard ITU este G.992.2. Este același ca în fig. 2-29, dar fără separator. Linia telefonică existentă este folosită fără nici o modificare. Singura diferență este că microfiltrul trebuie să fie introdus în fiecare mufă de telefon dintre telefon sau modem ADSL și fir. Microfiltrul pentru telefon este un filtru-trece-jos care elimină frecvențele mai mari de 3400 Hz; microfiltrul pentru ADSL este un filtru-trece-sus care elimină frecvențele sub 26 KHz. Oricum, acest sistem nu este la fel de fiabil ca și cel cu separator, deci G.lite poate fi folosit doar până la 1,5 Mbps (față de cei 8 Mbps de la ADSL cu separator). G.lite are nevoie oricum de separator la oficiul terminal, dar instalarea nu mai necesită mii de drumuri pentru intervenții la clienți.

ADSL e doar un standard pentru nivelul fizic. Ceea ce rulează la nivelurile superioare depinde de distribuitorul de Internet. Deseori, acesta alege ATM datorită posibilității ATM-ului de a satisface calitatea serviciului și datorită faptului că multe companii telefonice conțin ATM la baza rețelei.

Bucle locale fără fir

Din 1996 în SUA și ceva mai târziu în alte țări, companiile care doreau să intre în competiție cu puternicele companii locale de telefonie (fostele monopoluri), numite **ILEC** (**Incumbent LEC**, rom: LEC-uri de facto), sunt libere să o facă. Cei mai probabili candidați sunt companiile de telefoane pe distanță lungă (IXC-urile). Orice IXC care dorea să intre în telefonia locală trebuia să îndeplinească unele condiții. În primul rând, trebuie să cumpere sau să închirieze o clădire pentru primul oficiu terminal dintr-un oraș. În al doilea rând, trebuie să pună comutatoare de telefoane și alte echipamente în oficiul terminal, dispozitive care sunt puse în vânzare de diverși producători. În al treilea rând, trebuie tras un cablu cu fibră optică între oficiul terminal și cel mai apropiat oficiu, pentru ca noii consumatori să aibă acces la rețeaua națională. În al patrulea rând, trebuie să racoleze clienți, de obicei prin reclamă care anunță prețuri mai mici și servicii mai bune decât cele ale ILEC.

De aici începea parte grea. Să presupunem că apar câțiva clienți. Cum are de gând noua companie de telefoane locală, numită **CLEC** (**Competitive LEC**, rom: LEC competitivă) să conecteze telefoanele clienților la oficiul final proaspăt deschis? Obținerea drepturilor necesare și întinderea firelor sau a fibrei sunt acțiuni foarte costisitoare. Mute CLEC-uri au descoperit o alternativă la bucla tradițională de cablu torsadat: **WLL-ul (Wireless Local Loop**, rom: bucla locală fără fir).

Într-un anumit fel, un telefon fix care folosește o buclă locală fără fir seamănă cu un telefon mobil, dar sunt trei diferențe importante. Prima: clienții din bucla locală fără fir doresc deseori Internet de mare viteză, la viteze care să egaleze ADSL-ul. A doua: noul client nu are probabil nimic împotrivă ca un tehnician al CLEC să instaleze o antenă mare pe acoperișul său, direcționată către oficiul CLEC. A treia: utilizatorul nu se mută, eliminând toate problemele de mobilitate și timpii morți datorați celulelor despre care vom vorbi mai târziu în acest capitol. Și astfel se naște o nouă industrie: **fixă fără fir (fixed wireless)** (telefonie locală și servicii Internet oferite de CLEC pe o buclă locală fără fir).

Deși WLL și-a început activitatea semnificativă în 1998, trebuie să ne întoarcem în 1969 pentru a-i vedea originile. În acel an, FCC a alocat două canale TV (de 6MHz fiecare) pentru televiziunea educativă la 2,1GHz. În anii ce au urmat, au mai fost adăugate 31 de canale la 2,5GHz, cu un total de 198 MHz.

Televiziunea educativă nu a prins, iar în 1998 FCC a retras frecvențele și le-a alocat radioului bidirecțional. Au fost imediat acaparate de buclele locale fără fir. La aceste frecvențe, microundele au 10-12 cm. Au un domeniu de 50 km și penetrează vegetația și ploaia destul de bine. Cei 198 MHz noi din spectru au fot puși în uz pentru buclele locale fără fir ca un serviciu numit **MMDS** (**Multichannel Multipoint Distribution Service**, rom: serviciu de distribuție multicanal multipunct). MMDS poate fi privit ca un MAN (Rețea de acoperire Metropolitană), la fel ca și varul său LMDS (discutat mai jos).

Marele avantaj al acestui serviciu este că tehnologia este bine stabilită și echipamentele sunt disponibile. Dezavantajul este că lățimea de bandă disponibilă este modestă și trebuie folosită în comun de mulți utilizatori dintr-o arie geografică desul de mare.

Lățimea mică de bandă a MMDS a făcut din undele milimetrice o alternativă interesantă. La 28-31 GHz în SUA și 40 GHz în Europa nu se alocau frecvențe deoarece este dificil să construiești circuite integrate cu siliciu atât de rapide. Problema a fost rezolvată de inventarea circuitelor integrate cu galiu și arseniu, deschizându-se astfel banda milimetrică pentru radio-comunicații. FCC a răspuns cererii alocând 1,3 GHz unei noi bucle locale fără fir numită **LMDS (Local Multipoint Distribution Service**, rom: serviciu local de distribuție multipunct). Această alocare este cea mai mare alocare de lățime de bandă pe care a făcut-o FCC vreodată. O lățime similară este alocată și în Europa, dar la 40 GHz.

Modul de operare al LMDS este prezentat în fig. 2-30. În figură este prezentat un turn cu mai multe antene, fiecare fiind îndreptată într-o altă direcție. Cum razele milimetrice sunt foarte bine direcționate, fiecare antenă definește un sector, independent de celelalte. La această frecvență, raza de acțiune este de 2-5 km, ceea ce înseamnă că e nevoie de multe turnuri pentru a acoperi un oraș întreg.

Ca și ADSL, LMDS folosește o alocare de lățime de bandă asimetrică, favorizând canalul de recepție. Cu tehnologia curentă, fiecare sector poate avea 36 Gbps pentru recepție și 1 Mbps pentru transmisie, bandă folosită în comun de toți utilizatorii sectorului. Dacă fiecare utilizator activ descarcă trei pagini de 5 KB pe minut, utilizatorul ocupa in medie 200bps din spectru, ceea ce permite maxim 18000 de utilizatori activi pe sector. Totuși, pentru a se menține întârzierile la un nivel rezonabil, ar trebui să nu fie mai mult de 9.000 de utilizatori. Cu patru sectoare, ca în fig. 2-30, poate fi deservită o populație activă de 36.000 de locuitori. Presupunând ca unul din trei utilizatori este conectat în momentele de vârf, un singur turn cu patru antene poate deservi 100.000 de oameni într-o rază de 5 km față de turn. Aceste calcule au fost făcute de multe CLEC-uri, dintre care unele au ajuns la concluzia că făcând o investiție modestă în turnuri pentru unde milimetrice pot oferi utilizatorilor viteze comparabile cu cablul TV la un preț mai mic.

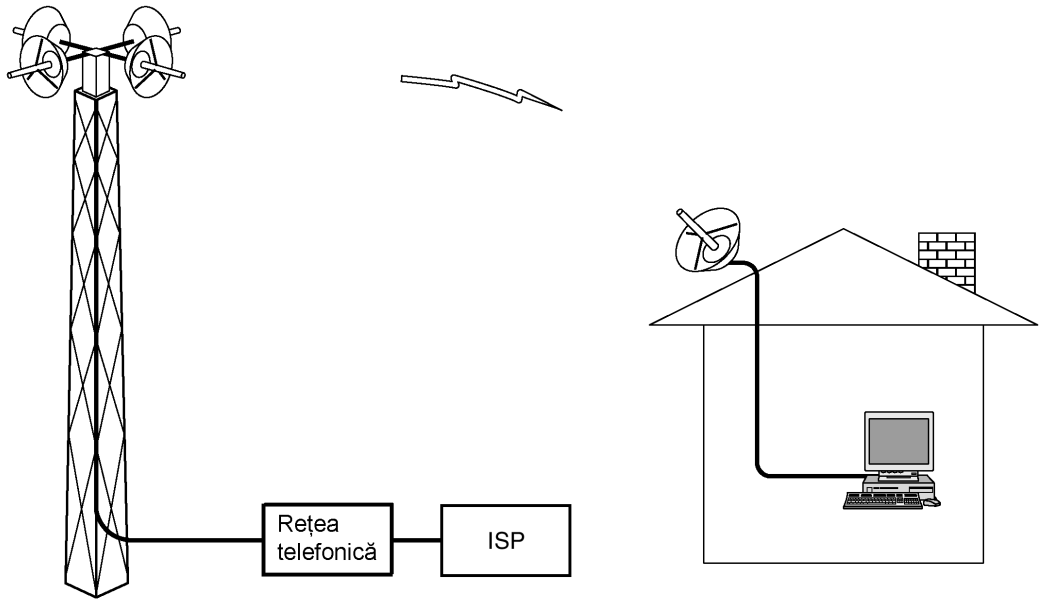


Fig. 2-30. Arhitectura unui sistem LMDS.

LMDS are totuși câteva probleme. Pentru început, undele milimetrice se propagă în linie dreaptă, deci trebuie să existe vizibilitate între antena de pe acoperiș și turn. O alta este că frunzele absorb aceste unde destul de bine, deci turnul trebuie să fie suficient de înalt pentru ca drumul până la antenă să treacă pe deasupra copacilor. Iar ceea ce pare un drum liber în decembrie s-ar putea să nu mai fie în iulie, când copacii sunt plini de frunze. Ploaia absoarbe și ea aceste unde. Totuși, erorile introduse de ploaie pot fi compensate cu un cod corector de erori sau prin mărirea puterii atunci când plouă. Oricum, serviciul LMDS e mult mai probabil să fie introdus mai întâi în ținuturile uscate, să zicem în Arizona, decât în Seattle.

Buclele locale fără fir nu au șanse să prindă dacă nu există standarde, pentru a încuraja vânzătorii de echipamente să producă și să asigure clienții că pot schimba CLEC-ul fără să fie nevoie să cumpere un alt echipament. Pentru a asigura acest standard, IEEE a întrunit un comitet numit 802.16 pentru a stabili standardul pentru LMDS. Standardul 802.16 a fost publicat în aprilie 2002. IEEE numește 802.16 **MAN fără fir** (**wireless MAN**). IEEE 802.16 a fost proiectat pentru telefonia digitală, acces la Internet, conectarea a două LAN-uri îndepărtate, emisie radio și TV, precum și pentru alte utilizări. Îl vom studia mai în detaliu în cap. 4.

2.5.4 Trunchiuri și multiplexare

Economiile rezultate din scalabilitate joacă un rol important în sistemul telefonic. Instalarea și întreținerea unor trunchiuri de bandă largă între două oficii de comutare costă cam tot atât cât instalarea și întreținerea unui trunchi de bandă joasă (costul provine de la săparea șanțului, nu de la firul de cupru sau de la fibra optică). În consecință, companiile telefonice au dezvoltat metode sofisticate pentru multiplexarea mai multor convorbiri pe aceeași magistrală fizică. Aceste metode de multiplexare se pot împărți în două categorii principale: **FDM** (**Frequency Division Multiplexing**, rom: multiplexare cu

divizare în frecvență) și **TDM** (**Time Division Multiplexing**, rom: multiplexare cu divizare în timp). La FDM, spectrul de frecvență este împărțit în mai multe canale logice, fiecare utilizator având drepturi exclusive asupra unei anumite benzi de frecvență. La TDM, utilizatorii își așteaptă rândul (în mod repetat, circular), fiecare utilizator obținând întreaga bandă de frecvență pentru o scurtă perioadă.

Difuzarea radio AM ilustrează ambele metode de multiplexare. Spectrul alocat este de aproape 1MHz, aproximativ între 500 și 1500 de KHz. Pentru diferite canale logice (stații) sunt alocate frecvențe diferite. Fiecare canal logic operează într-un anumit domeniu al spectrului, distanțele între canale fiind destul de mari pentru a preveni interferența. Acest sistem este un exemplu de multiplexare prin divizarea frecvenței. În plus (în unele țări), stațiile individuale au două subcanale logice: muzică și publicitate. Acestea două alternează în timp pe aceeași frecvență, la început muzică și după un timp o secvență de reclame, apoi din nou muzică și așa mai departe. Această situație se numește multiplexare prin divizare în timp.

În continuare, vom studia multiplexarea prin divizarea în frecvență. După aceea vom analiza cum se poate aplica FDM fibrelor optice (multiplexare prin divizarea lungimii de undă). Apoi ne vom întoarce la TDM, iar în final vom studia un sistem TDM avansat folosit în fibrele optice (SONET).

Multiplexarea prin divizarea în frecvență

Fig. 2-31 ne prezintă cum sunt multiplexate trei canale de bandă vocală folosind FDM. Filtrele limitează lărgimea de bandă folosită la 3100 de Hz pe canal de bandă vocală. Atunci când sunt multiplexate împreună mai multe canale, fiecărui canal îi sunt alocați 4000 de Hz, astfel încât canalele să fie bine separate. Mai întâi, canalele de voce sunt deplasate în frecvență, fiecare cu o valoare diferită. Apoi ele pot fi combinate, deoarece nu există două canale care să ocupe aceeași zonă a spectrului. Este de remarcat că, deși există spații (spații de gardă) între canale, poate să apară o suprapunere între canalele adiacente, deoarece filtrele nu au marginile abrupte. Această suprapunere înseamnă că un semnal puternic la capătul unui canal va fi simțit în canalul adiacent ca un zgomot non-termic.

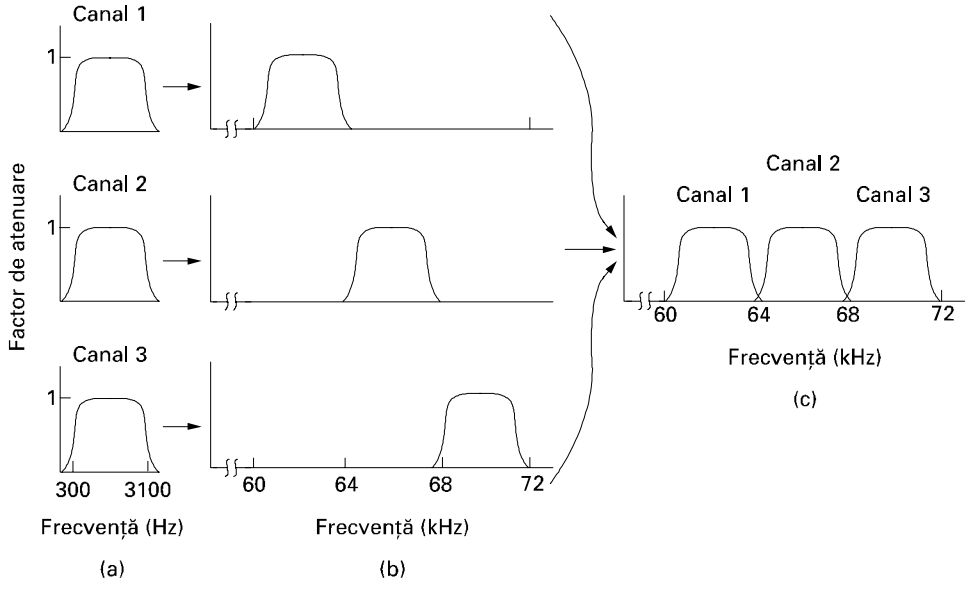


Fig. 2-31. Mutliplexarea prin divizare în frecvență. (a) Banda de frecvență inițială. (b) Banda deplasată în frecvență. (c) Canalul multiplexat.

Schemele FDM folosite pe glob sunt, până la un anumit nivel, standardizate. Un standard foarte folosit este dat de 12 canale vocale a 4000 Hz multiplexate în banda de 60 până la 108 KHz. Această unitate este numită **grup**. Banda 12 - 60 KHz este uneori folosită de un alt grup. Multe companii oferă clienților un serviciu de linie închiriată de 48 până la 56 Kbps, bazat pe un grup. Cinci grupuri (60 de canale vocale) formează un **super-grup**. Următoarea unitate este un **master-grup**, care este format din cinci super-grupuri (în standardul CCITT) sau zece super-grupuri (sistemul Bell). Există și alte standarde, care cuprind până la 230.000 canale vocale.

Multiplexarea prin divizarea lungimii de undă

Pentru canalele de fibră optică se utilizează o alternativă a multiplexării prin divizarea în frecvență. Aceasta se numește **WDM (Wavelength Division Multiplexing**, rom: multiplexarea prin divizarea lungimii de undă). Principiul de bază pentru WDM pe fibre este prezentat în fig. 2-32. Aici se întâlnesc patru fibre la nivelul unui combinator optic, fiecare cu energia și lungimea de undă proprie. Cele patru raze sunt combinate într-o singură fibră comună pentru a fi transmise către o destinație depărtată. La destinație, raza este despărțită în atâtea fibre câte au fost inițial. Fiecare fibră de la destinație conține un mic filtru special construit, care filtrează toate lungimile de undă mai puțin una. Semnalele rezultate pot fi rutate către destinație sau recombinate în diferite feluri pentru transmisii multiplexate ulterioare.

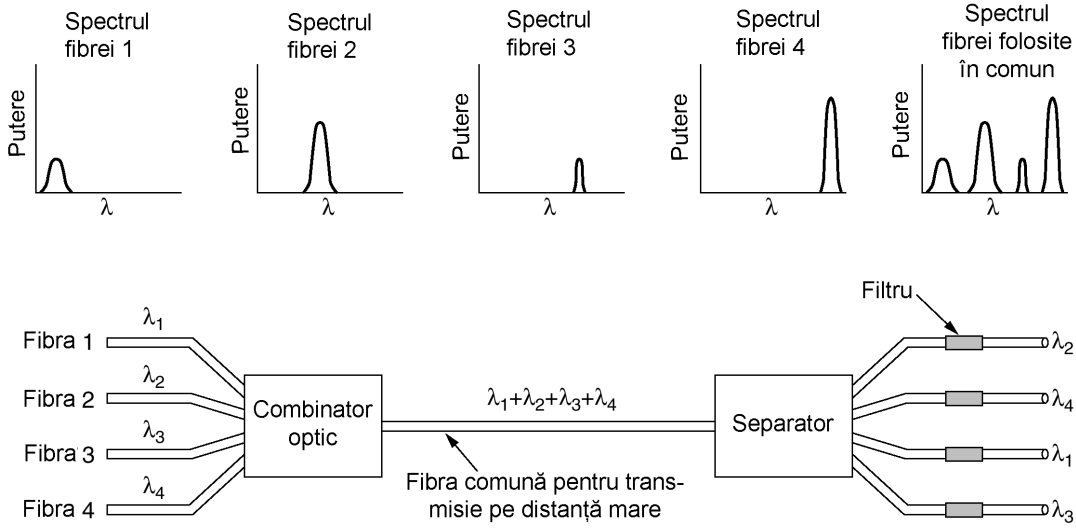


Fig. 2-32. Multiplexarea prin divizarea lungimii de undă

Aici nu este de fapt nimic nou. Este tot multiplexare prin divizarea în frecvență, aplicată la frecvențe foarte mari. Atâta timp cât fiecare canal are propriul domeniu de frecvență (lungime de undă) și toate aceste domenii sunt disjuncte, ele pot fi multiplexate împreună pe o fibră de distanță mare. Singura diferență față de FDM prezentat anterior este aceea că sistemul optic, folosind o rețea de difracție, este total pasiv și, de aceea, foarte sigur.

Tehnologia WDM a progresat extrem de rapid, umbrind chiar și tehnologia calculatoarelor. WDM a fost inventată în jurul anului 1990. Primul sistem comercial avea opt canale, fiecare de 2,5 Gbps. În 1998 existau deja pe piață sisteme cu 40 de canale de 2,5 Gbps fiecare. În 2001, existau pro-

produse cu 96 de canale de 10 Gbps fiecare, deci un total de 960 Gbps. Această lățime de bandă este suficientă pentru a transmite filme întregi într-o secundă (format MPEG-2). În laboratoare se lucrează deja la sisteme cu 200 de canale. Atunci când numărul de canale este foarte mare și lungimile de undă sunt foarte apropiate, de exemplu la 0,1 nm, sistemul mai este numit și **DWDM (Dense WDM**, rom: WDM densă).

Este de remarcat că WDM este foarte populară datorită faptului că energia pe o singură fibră este, de obicei, cuprinsă într-o bandă de numai câțiva gigaherți, deoarece în acest moment este imposibilă conversia mai rapidă de la mediul electric la cel optic. Prin folosirea în paralel a mai multor canale, de lungimi de undă diferite, lățimea de bandă crește liniar cu numărul de canale. Din moment ce banda de frecvență a unei singure fibre este de aproximativ 25.000 GHz (vezi fig. 2-6), există posibilitatea de a se ajunge la 2500 de canale de 10Gbps fiecare, chiar și la o rată de 1 bit/Hz (fiind posibile de asemenea și rate mai mari).

O altă direcție de dezvoltare o reprezintă toate amplificatoarele optice. Înainte vreme, la fiecare 100km era obligatorie separarea tuturor canalelor și conversia fiecărui semnal optic într-un semnal electric; fiecare semnal era amplificat separat înainte de a se face conversia inversă, din semnal electric în semnal optic, urmată de recombinarea semnalelor optice. În prezent, toate amplificatoarele optice pot regenera întregul semnal odată la 1000km fără a fi nevoie de conversii multiple între semnalele electrice și optice.

În exemplul din fig. 2-32, avem un sistem cu lungimi de undă fixe. Unii biți din fibra de intrare 1 trec în fibra de ieșire 3, biți din fibra de intrare 2 trec în fibra de ieșire 1 etc. Oricum, este posibil să construim și sisteme WDM comutate. În cazul unui astfel de dispozitiv, filtrele de ieșire sunt reglabile folosind interferometre Fabry-Perot sau Mach-Zender. Pentru informații suplimentare despre WDM și aplicațiile sale în comutarea pachetelor în Internet vezi (Elmirghani și Mouftah, 2000; Hunter și Andonovic, 2000; Listani și alții, 2001)

Multiplexarea prin divizarea în timp

Tehnologia WDM este minunată, dar există încă multe fire din cupru în rețeaua telefonică, deci să revenim un timp la discuțiile referitoare la acestea. Cu toate că FDM este folosită încă pe firele de cupru sau pe canalele de microunde, ea necesită circuite analogice și nu poate fi realizată de un calculator. Dimpotrivă, TDM-ul poate fi în întregime tratat de electronica digitală, devenind mult mai răspândit în ultimii ani. Din păcate, TDM-ul poate fi folosit numai pentru date digitale. Deoarece buclele locale produc semnale analogice, este necesară o conversie analogic – digital în oficiul final, unde toate buclele locale individuale se întâlnesc pentru a forma o magistrală.

Vom arunca acum o privire asupra modului în care mai multe semnale vocale analogice sunt digitizate și sunt reunite pentru a forma o singură magistrală digitală. Datele trimise de un calculator prin modem sunt tot analogice, deci următoarea descriere se aplică și pentru acestea. Semnalele analogice sunt digitizate în oficiul final de un echipament numit **codec** (codificator/decodificator), operație care are ca rezultat o serie de numere de 8 biți. Codec-ul realizează 8000 de eșantioane pe secundă (125 μsec/eșantion), deoarece teorema lui Nyquist spune că aceasta rată este suficientă pentru a capta toată informația de pe canalul telefonic de 4 KHz. La o rată mai mică de eșantionare, o parte din informație ar putea fi pierdută; la o rată mai mare, nu se furnizează nici un fel de informație suplimentară. Această tehnică se numește **PCM** (**Pulse Code Modulation**, rom: modularea în cod de impulsuri). PCM constituie nucleul sistemelor telefonice moderne. Ca o consecință, practic toate intervalele de timp dintr-un sistem telefonic sunt multipli de 125 μsec.

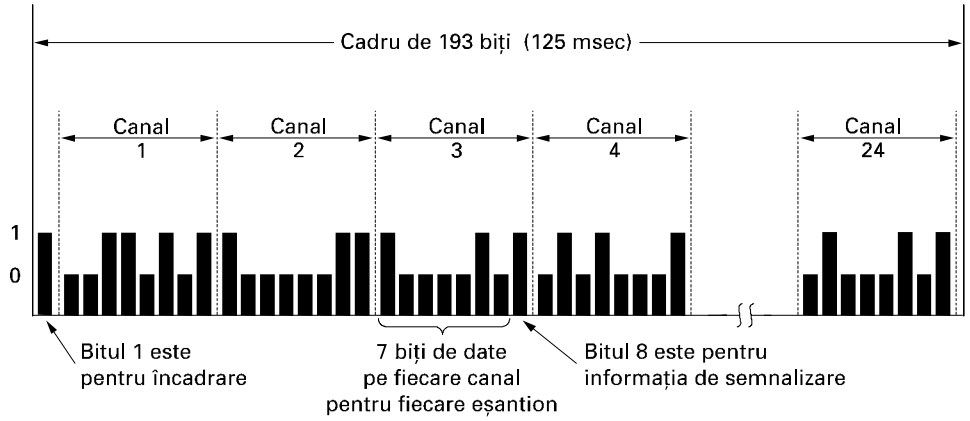


Fig. 2-33. Purtătoarea T1 (1.544 Mbps).

Când transmisia digitală a început să devină o tehnologie accesibilă, CCITT nu a fost capabil să ajungă la o soluție cu privire la un standard pentru PCM. În consecință, acum există o varietate de scheme incompatibile, folosite în diverse țări din întreaga lume.

O metodă răspândită în America de Nord și Japonia o reprezintă purtătoarea **T1**, prezentată în fig. 2-33. (Din punct de vedere tehnic, formatul se numește DS1 iar purtătoarea se numește T1, dar noi nu vom face această deosebire subtilă aici). Purtătoarea T1 constă din 24 de canale vocale multiplexate împreună. De obicei, cele 24 de semnale analogice sunt eșantionate pe rând, iar fluxul analogic rezultat este trecut prin codec, în loc să se utilizeze 24 de codec-uri separate a căror semnale de ieșire să compună ieșirea digitală. Fiecare canal din cele 24 va introduce 8 biți în secvența de la ieșire. Șapte biți reprezintă date, iar unul este de control, rezultând 7x8000 = 56.000 bps de date și 1x8000 = 8.000 bps de informație de semnalizare pe fiecare canal.

Un cadru constă din 24x8 = 192 biți, plus un bit suplimentar pentru încadrare, rezultând 193 de biți la fiecare 125 μsec. De aici rezultă o viteză de transfer a datelor de 1.544 Mbps. Bitul 193 este folosit pentru sincronizarea cadrelor, și urmează un model de forma 0101010101... . În mod normal, receptorul verifică în continuu acest bit, pentru a fi sigur că nu a pierdut sincronizarea. Dacă într-adevăr a pierdut tactul, receptorul poate căuta după acest șablon pentru a se resincroniza. Clienții analogici nu pot genera un semnal conform cu acest șablon, deoarece el corespunde unei unde sinusoidale la 4000 Hz, care ar fi eliminată prin filtrare. Clienții digitali pot, desigur, să genereze un semnal care să respecte acest șablon, dar șansele apariției lui sunt mici atunci când un cadru este pierdut. Atunci când un sistem T1 este folosit integral pentru date, doar 23 de canale sunt folosite pentru date. Canalul 24 este folosit pentru un șablon de sincronizare special, care permite reluarea mai rapidă a funcționării în cazul în care un cadru este pierdut.

În cele din urmă, când CCITT a ajuns la o înțelegere, a considerat că a folosi 8000 bps de informație de semnalizare este o exagerare, așa că standardul de 1.544 Mbps se bazează pe date de 8 biți în loc de 7 biți; aceasta înseamnă că semnalul analogic este codificat folosind 256 de niveluri discrete în loc de 128. Există două situații, incompatibile. În **semnalizarea prin canal comun** (**common-channel signaling**), bitul suplimentar (care este atașat mai degrabă în urma, și nu în fața cadrului de 193 de biți) are valorile 1010101010... în cadrele impare și conține informații de semnalizare pentru toate canalele în cadrele pare.

În cealaltă alternativă, **semnalizarea pe canale asociate** (**channel-asssociated signaling**), fiecare canal are propriul său subcanal de semnalizare. Un subcanal privat este constituit prin alocarea unui bit pentru semnalizare (dintre cei 8 biți utili) la fiecare 6 cadre; astfel, 5 din 6 eșantioane sunt de 8 biți lungime, iar ultimul este de doar 7 biți lungime. De asemenea, CCITT a recomandat o purtătoare PCM de 2.048 Mbps numită **E1**. Această purtătoare are 32 de eșantioane de 8 biți, împachetate în cadrul de bază de 125 μsec. Treizeci dintre aceste canale sunt folosite pentru informații, iar celelalte două sunt folosite pentru semnalizare. Fiecare grup de 4 cadre asigură 64 de biți de semnalizare, dintre care jumătate sunt folosiți pentru semnalizarea pe canalul asociat și jumătate pentru sincronizarea cadrelor sau pentru alte operații, în funcție de țările unde sunt utilizate. Purtătoarea E1 de 2.048 Mbps este larg folosită în afara Americii de Nord și a Japoniei.

Odată ce semnalul vocal a fost codificat, este tentant să folosim metode statistice pentru a reduce numărul de biți necesari pentru fiecare canal. Aceste tehnici sunt potrivite nu numai pentru codificarea vorbirii, ci și pentru digitizarea oricărui semnal analogic. Toate metodele de compactare se bazează pe principiul că semnalul se modifică relativ încet în comparație cu frecvența de eșantionare, deci multe dintre informațiile codificate pe 7 sau 8 biți sunt redundante.

Metoda, numită **modulare diferențială în cod de impulsuri** (**differential pulse code modulation**), constă în furnizarea la ieșire nu a amplitudinii digitizate, ci a diferenței dintre valoarea curentă și cea anterioară. Deoarece sunt puțin probabile salturi mai mari de ±16 pe o scală de 128, ar putea fi suficienți 5 biți în loc de 7. Dacă semnalul are salturi prea bruște, logica de codificare poate necesita mai multe perioade de eșantionare pentru „a prinde din urmă” aceste variații. În cazul vorbirii, eroarea introdusă poate fi ignorată.

O variantă a acestei metode de compactare impune ca fiecare valoare eșantionată să difere de precedenta prin +1 sau -1. În aceste condiții, este transmis un singur bit, care indică dacă nivelul curent este superior sau inferior nivelului precedent. Această tehnică, numită **modulare delta**, este ilustrată în fig. 2-34. La fel ca toate tehnicile de compactare care presupun schimbări mici de nivel între eșantioane consecutive, dacă sistemul se schimbă prea brusc, modularea delta poate eșua, așa cum se arată în figură. Când se întâmplă acest lucru, informația se pierde.

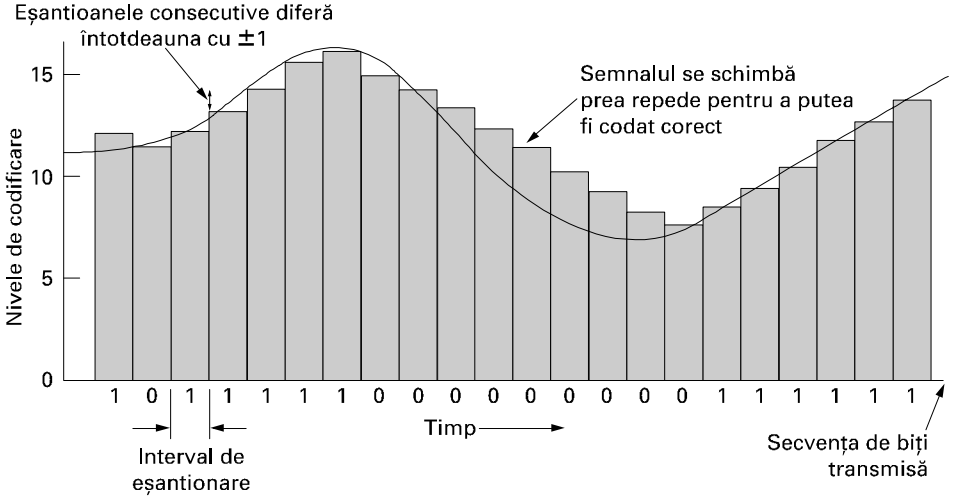


Fig. 2-34. Modularea Delta.

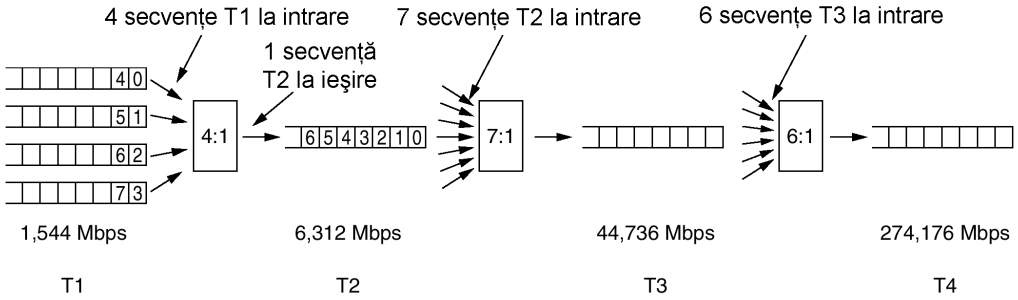


Fig. 2-35. Multiplexarea secvențelor T1 în purtătoare cu capacitate mai mare.

O îmbunătățire pentru PCM diferențial se obține dacă se extrapolează câteva valori precedente pentru a prezice noua valoare și apoi se codifică diferența dintre semnalul actual și cel prevăzut. Desigur, atât transmițătorul cât și receptorul trebuie să folosească același algoritm de predicție. O astfel de schemă se numește **codificare predictivă**. Această tehnică este utilă, pentru că reduce dimensiunea datelor care trebuie codificate și, prin urmare, reduce dimensiunea datelor care trebuie transmise.

Multiplexarea prin divizare în timp permite ca mai multe purtătoare T1 să fie multiplexate împreună în purtătoare de un grad mai înalt. Fig. 2-35 arată cum se poate face acest lucru. În stânga se văd patru canale T1 multiplexate într-un canal T2. Multiplexarea la T2 și peste T2 se face bit cu bit și nu octet cu octet, ca în cazul celor 24 de canale care constituie un cadru T1. Patru secvențe T1 la 1.544 Mbps ar trebui să genereze 6.176 Mbps, dar T2 transmite de fapt la 6.312 Mbps. Biții suplimentari sunt folosiți la încadrare și la recuperare, în cazul în care purtătoarea este pierdută. T1 și T3 sunt folosite pe scară largă de consumatori, în timp ce T2 și T4 sunt folosite numai în cadrul sistemului de telefonie, deci nu sunt foarte cunoscute.

La nivelul următor, șapte T2 sunt combinate pentru a forma o secvență T3. Apoi șase T3 sunt grupate pentru a forma o secvență T4. La fiecare pas, se adăugă, pentru încadrare și recuperare, o mică supraîncărcare în cazul în care sincronizarea dintre transmițător și receptor se pierde.

Așa cum există neînțelegeri privind purtătoarea de bază, între Statele Unite și restul lumii, tot așa există neînțelegeri privitoare la modul în care se face multiplexarea în purtătoarele de bandă mai largă. Ierarhia implementată în S.U.A., care este făcută prin grupuri de 4, 7 și 6 secvențe, nu s-a impus ca un standard, standardul CCITT multiplexând 4 secvențe în una singură la fiecare nivel. De asemenea, încadrarea și recuperarea datelor se fac diferit. Ierarhia CCITT cu 32, 128, 512, 2048 și 8192 de canale funcționează la viteze de 2.048, 8.848, 34.304, 139.264 și 565.148 Mbps.

SONET / SDH

La apariția fibrelor optice, fiecare companie telefonică avea propriul sistem optic TDM. După ce AT&T-ul a fost divizat în 1984, companiile telefonice locale au fost obligate să se conecteze la diverse companii de telecomunicații pe distanțe mari, fiecare companie având sisteme TDM diferite; a devenit astfel evidentă necesitatea unei standardizări. În 1985, Bellcore, divizia de cercetare a RBOC, a început să lucreze la un nou standard, numit **SONET** (**Synchronous Optical NETwork**, rom: rețea optică sincronă). Mai târziu s-a alăturat și CCITT, fapt care s-a materializat în 1989 prin standardul SONET și printr-un set paralel de recomandări CCITT (G.707, G.708 și G.709). Recomandările CCITT sunt numite **SDH** (**Synchronous Digital Hierarchy**, rom: ierarhie digitală sincronă), dar diferă de SONET numai în mică măsură. Practic, aproape tot traficul pe distanțe mari din Statele Unite și

cea mai mare parte a traficului din alte zone folosește acum trunchiuri cu SONET pe nivelul fizic. Pentru informații suplimentare vezi (Bellamy 2000; Goralski, 2000; și Shepard, 2001).

Proiectul SONET a urmărit patru obiective principale. Primul și cel mai important, SONET trebuia să permită conlucrarea mai multor companii de telecomunicații. Pentru atingerea acestui obiectiv a fost necesară definirea unui standard comun de codificare a semnalului care să facă referire la lungimea de undă, la sincronizare, la structura cadrelor etc.

În al doilea rând, erau necesare câteva metode de a unifica sistemele digitale din S.U.A., Europa și Japonia, toate bazându-se pe canale PCM de 64 Kbps, însă combinate diferit și devenind totodată incompatibile între ele.

În al treilea rând, SONET trebuia să permită multiplexarea mai multor canale digitale. Atunci când a fost elaborat SONET, cea mai rapidă purtătoare digitală utilizată efectiv pe scară largă în Statele Unite era T3, la 44.736 Mbps. T4 era definit, dar nu era atât de folosit, iar deasupra vitezei lui T4 nu era definit nimic. O parte a misiunii SONET era să ridice ierarhia la nivel de gigabiți/sec și chiar mai mult. Era de asemenea necesară o modalitate standard de multiplexare a canalelor mai lente într-un canal SONET.

În al patrulea rând, SONET trebuia să asigure suportul de operare, administrare și întreținere (OAM - Operations, Administration, Maintenance). Sistemele precedente nu au realizat acest lucru foarte bine.

O decizie anterioară era să se facă din SONET un sistem TDM tradițional, în care toată banda de frecvență a fibrei optice atribuită unui singur canal să conțină diferite intervale de timp pentru subcanale diferite. Astfel, SONET este un sistem sincron. El este controlat de un ceas principal, cu o eroare de aproximativ 10^{-9}. Biții sunt transmiși pe o linie SONET la intervale extrem de precise, controlate de ceasul principal. Atunci când comutarea celulelor a fost propusă ca bază pentru ATM în bandă largă, faptul că ea permitea sosirea neregulată a celulelor a determinat etichetarea sa ca mod de transfer *asincron* (ATM), pentru a contrasta astfel cu operațiile sincrone ale SONET-ului. Cu SONET, transmițătorul și receptorul sunt legați de un ceas comun; cu ATM nu sunt.

Cadrul de bază SONET este un bloc de 810 octeți, lansat la fiecare 125 μsec. Deoarece SONET este sincron, cadrele sunt emise, chiar dacă nu există date utile de transmis. Rata de 8000 cadre/sec coincide cu viteza de eșantionare a canalelor PCM folosite în sistemele telefonice digitale.

Cadrele SONET de 810 octeți sunt cel mai bine descrise prin matrice cu 90 de coloane și 9 rânduri. Cei 8x810 = 6480 biți ai unui cadru sunt transmiși de 8000 de ori pe secundă, la o viteză de transfer a datelor de 51,84 Mbps. Acesta este canalul de bază SONET și este numit **STS-1** (**Synchronous Transport Signal,** rom: semnal sincron de transport). Toate trunchiurile SONET sunt multiple de STS-1.

Primele trei coloane ale fiecărui cadru sunt rezervate pentru informația de administrare a sistemului, așa cum este prezentat în fig. 2-36. Primele trei rânduri conțin informația suplimentară (eng: overhead) pentru secțiune; următoarele șase conțin informația suplimentară pentru linie. Informația suplimentară pentru secțiune este generată și verificată la începutul și la sfârșitul fiecărei secțiuni, în timp ce informația suplimentară pentru linie este generată și verificată la începutul și la sfârșitul fiecărei linii.

Un transmițător SONET trimite cadre succesive de 810 octeți, fără pauze între ele, chiar și atunci când nu există date de transmis (situație în care trimite cadre fără semnificație). Din punct de vedere al receptorului, informația este doar un șir continuu de biți, deci cum să știe de unde începe un cadru? Răspunsul este că primii doi octeți ai fiecărui cadru conțin un șablon fix pe care receptorul îl caută. Dacă găsește șablonul în același loc la mai multe cadre consecutive, presupune ca s-a sincronizat cu transmițătorul. Teoretic, un utilizator ar putea trimite acest șablon în informația utilă din ca-

dru, dar practic acest lucru nu poate fi realizat din cauza multiplexării mai multor utilizatori în același cadru și din alte motive.

Restul de 87 de coloane conțin 87 x 9 x 8 x 8000 = 50.112 Mbps de date ale utilizatorului. Totuși, datele utilizatorului, numite **SPE** (**Synchronous Payload Envelope**, rom: înveliș pentru informație utilă sincronă) nu încep întotdeauna cu rândul 1, coloana 4. SPE poate începe oriunde în interiorul cadrului. Un pointer către primul octet este conținut în primul rând al informației suplimentare pentru linie. Prima coloană din SPE este informația suplimentară pentru cale (adică un antet pentru protocolul subnivelului de conexiune capăt-la-capăt).

Posibilitatea ca SPE să înceapă oriunde în cadrul SONET și chiar să se întindă pe două cadre, așa cum este prezentat în fig. 2-36, conferă sistemului un grad suplimentar de flexibilitate. De exemplu, dacă la sursă ajung date în timp ce se construiește un cadru SONET gol, aceste date pot fi inserate în cadrul curent în loc să fie reținute până la începutul următorului cadru.

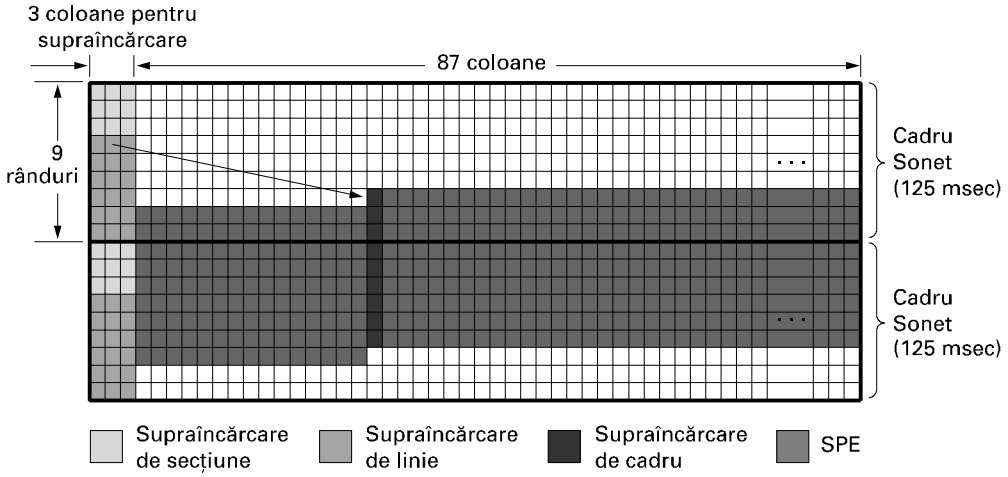


Fig. 2-36. Două cadre SONET succesive.

Ierarhia de multiplexare SONET este prezentată în fig. 2-37. Au fost definite viteze de la STS-1 până la STS-192. Purtătoarea optică pentru STS-*n* este numită OC-*n* și este identică bit cu bit cu STS-*n*, cu excepția unui rearanjări a biților folosită pentru sincronizare. Numele SDH sunt diferite și încep de la OC-3, deoarece sistemele bazate pe standardele CCITT nu au o viteză de transfer apropriată de 51.84 Mbps. Purtătorul OC-9 este prezent, deoarece este apropriat de viteza de transfer a unui trunchi de mare viteză folosit în Japonia. OC-18 și OC-36 sunt folosite în Japonia. La calculul vitezei de transfer a datelor sunt incluse toate informațiile suplimentare. La calculul vitezei de transfer SPE se exclud informațiile suplimentare pentru linie și secțiune. Pentru a determina viteza de transfer a datelor utile este exclusă orice informație suplimentară, fiind luate în considerație numai cele 86 de coloane puse la dispoziție pentru datele utile.

Ca un comentariu suplimentar, rețineți că atunci când o purtătoare, cum este OC-3, nu este multiplexată, dar transportă date de la o singură sursă, se adaugă la notație litera *c* (de la concatenare), astfel că OC-3 indică o purtătoare de 155,52 Mbps care constă din trei purtătoare OC-1 diferite, în timp ce OC-3c indică o secvență de date de la o singură sursă la 155,52 Mbps. Cele trei secvențe OC-1 dintr-o secvență OC-3c sunt întrețesute pe coloane, prima coloană din secvența 1, apoi coloana din secvența 2, apoi coloana 1 din secvența 3, urmată de coloana 2 din secvența 1 și așa mai departe, ceea ce conduce la un cadru de 270 de coloane și 9 rânduri.

SONET		SDH	Rata de date (Mbps)		
Electric	Optic	Optic	Totală	SPE	Client
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC -3	STM-1	155.52	150.336	148.608
STS-9	OC -9	STM-3	466.56	451.008	445.824
STS-12	OC -12	STM-4	622.08	601.344	594.432
STS-18	OC -18	STM-6	933.12	902.016	891.648
STS-24	OC -24	STM-8	1244.16	1202.688	1188.864
STS-36	OC -36	STM-12	1866.24	1804.032	1783.296
STS-48	OC -48	STM-16	2488.32	2405.376	2377.728
STS-192	OC -192	STM-64	9953.28	9621.504	9510.912

Fig. 2-37. Ratele de multiplexare pentru SONET și SDH.

2.5.5 Comutarea

Din punctul de vedere al unui inginer obișnuit specialist în telefonie, sistemul telefonic se împarte în două: domeniul exterior (buclele locale și trunchiurile, deoarece ele sunt în afara oficiilor de comutare) și domeniul interior (comutatoarele). Tocmai am aruncat o privire asupra domeniului exterior. Acum a venit timpul să examinăm domeniul interior.

În sistemul telefonic se utilizează două tehnici de comutare diferite: comutarea de circuite și comutarea de pachete. Vom face în continuare câte o scurtă introducere pentru fiecare dintre cele două tehnici. Apoi vom studia în detaliu comutarea de circuite, acesta fiind modul de funcționare actual al sistemul telefonic. Comutarea de pachete va fi studiată în detaliu în capitolele care urmează.

Comutarea de circuite

Atunci când formezi – tu sau calculatorul tău – un număr de telefon, echipamentele de comutare din sistemul telefonic caută o cale fizică între telefonul tău și telefonul apelat. Această tehnică se cheamă **comutare de circuite** și este prezentată schematic în fig. 2-38(a). Fiecare dintre cele 6 dreptunghiuri reprezintă un oficiu de comutare al companiei de telecomunicații (oficiu final, oficiu de taxare etc.). În acest exemplu, fiecare oficiu are trei linii de intrare și trei linii de ieșire. Atunci când o cerere trece prin oficiu, se stabilește (conceptual) o legătură între linia de pe care a venit cererea și una din liniile de ieșire; aceste legături sunt reprezentate în figură de liniile punctate.

În primele zile ale telefoniei, legătura era făcută de un operator care conecta un cablu în mufele de intrare și de ieșire. De fapt, există și o povestioară simpatică legată de invenția echipamentelor de comutare automată a circuitelor. Aceste echipamente au fost inventate în secolul XIX de un proprietar al unei firme de Pompe Funebre din Missouri, pe nume Almon B. Strowger. Puțin timp după ce telefonul a fost inventat, când cineva a murit, unul dintre supraviețuitori ar fi sunat operatorul orașului și spunând: „Vă rog, faceți-mi legătura cu o firmă de Pompe Funebre”. Din păcate pentru domnul Strowger, existau două firme de Pompe Funebre în oraș, iar proprietarul celeilalte era soțul operatoarei telefonice a orașului. Domnul Strowger și-a dat repede seama că fie va inventa echipamentul telefonic de comutare automată, fie va renunța la afacere. A ales prima opțiune. Timp de 100 de ani, echipamentele de comutare a circuitelor au fost cunoscute în toată lumea drept cutia Strowger. (Istoria nu a consemnat însă dacă operatoarea de comutare, rămasă șomeră, a obținut ulterior un post de operator de informații, răspunzând la întrebări de genul „Care este numărul de telefon al unei firme de Pompe Funebre ?”).

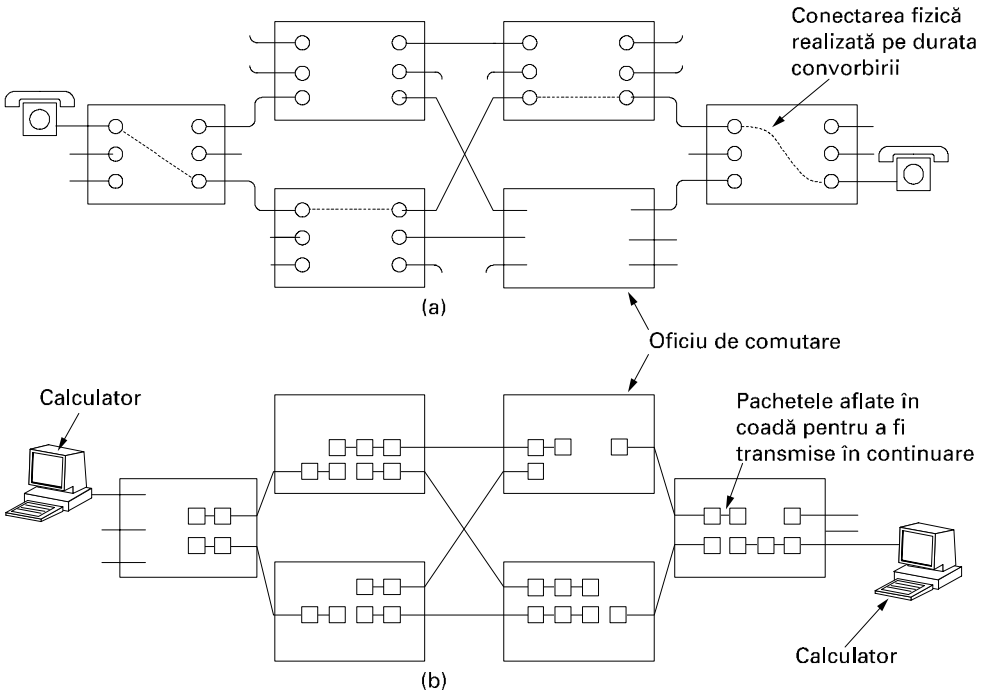


Fig. 2-38. (a) Comutare de circuite. (b) Comutare de pachete

Modelul prezentat în fig. 2-39(a) este, desigur, foarte simplificat, deoarece unele porțiuni din „drumul de cupru" între cele două telefoane pot fi, de fapt, legături prin microunde pe care sunt multiplexate mii de convorbiri. În orice caz, ideea de bază este validă: odată ce apelul a fost stabilit, există o cale dedicată între cele două capete și va continua să existe până când convorbirea se termină.

Alternativa la comutarea circuitelor este comutarea de pachete, descrisă în fig. 2-38(b). În cazul acestei tehnologii, se vor trimite pachete individuale la cerere, fără ca o cale dedicată să fie construită în prealabil. Fiecare pachet trebuie să-și găsească singur drumul către destinație.

O proprietate importantă a comutării de circuite este nevoia de a stabili o cale de la un capăt la celălalt, înainte ca datele să poată fi transmise. Intervalul de timp dintre momentul formării numărului și până când se aude sunând telefonul apelat poate ajunge ușor la 10 sec, chiar mai mult pe distanțe mari sau în cazul convorbirilor internaționale. În acest interval de timp, sistemul telefonic caută un drum prin cupru, după cum e prezentat în fig. 2-39(a). De remarcat că, înainte ca transmisia de date să poată începe, semnalul de apel trebuie să se propage până la destinație. Pentru multe aplicații pe calculator (de ex. verificarea creditului la punctul de vânzare), se dorește evitarea perioadelor lungi de setare.

Ca o consecință a căii rezervate dintre cele două parți care vorbesc, odată ce conexiunea a fost realizată, singura întârziere a datelor este dată de timpul de propagare a semnalului electromagnetic, de aproximativ 5 ms la 1000 Km. Totodată, ca o consecință a prestabilirii traseului, pericolul de congestie dispare – aceasta înseamnă că, odată ce apelul s-a efectuat, nu vei mai primi semnalul „ocupat". Desigur, poți primi semnalul „ocupat" înainte de a stabili legătura, datorită imposibilității de comutare sau a capacității insuficiente a trunchiului.

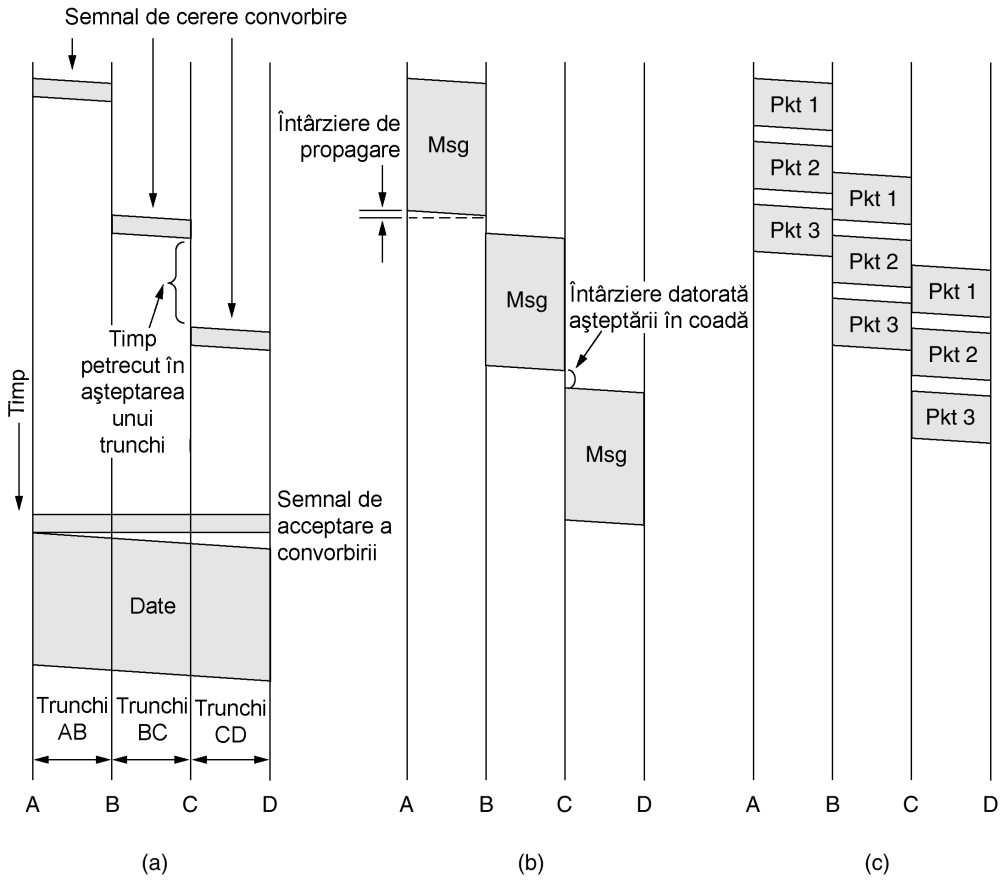


Fig. 2-39. Apariția evenimentelor în (a) comutarea de circuite. (b) comutarea de mesaje. (c) comutarea de pachete

Comutarea de mesaje

O alternativă la strategia de comutare de circuite este **comutarea de mesaje**, prezentată în fig. 2-39(b). Atunci când se utilizează acest tip de comutare, nu se stabilește de la început o cale între apelant și apelat. În schimb, atunci când apelantul are de transmis un bloc de date, acesta este memorat în primul oficiu de comutare (ruter) și este retransmis mai târziu, pas cu pas. Fiecare bloc este recepționat în întregime, verificat pentru a detecta eventualele erori și apoi retransmis. După cum a fost menționat în cap. 1, o rețea care folosește această tehnică se numește rețea **memorează-și-retransmite**.

Primul sistem de telecomunicație electromecanic se baza pe comutarea de mesaje și a fost folosit pentru telegrame. Mesajul era perforat pe o bandă de hârtie la oficiul de transmisie, bandă care era apoi citită și transmisă pe o linie de comunicație către următorul oficiu de pe traseu, unde era perforat pe o altă bandă de hârtie. Un operator de acolo rupea banda de hârtie și o citea pe unul din multele cititoare de bandă hârtie, câte unul pentru fiecare magistrală de ieșire. Un astfel de oficiu de comutare era numit **oficiu de tocare a benzilor** (**torn tape office**). Benzile de hârtie nu mai se folosesc și nici comutarea mesajelor, deci nu vom mai discuta acest subiect în această carte.

Comutarea de pachete

În cazul comutării de mesaje, nu există nici o limită a dimensiunii blocului, ceea ce înseamnă că ruterele (din sistemele moderne) necesită discuri pentru memorarea unor blocuri mari. De asemenea, acest lucru înseamnă că un singur bloc poate ocupa o linie ruter-ruter minute întregi, comutarea de mesaje nefiind utilă pentru traficul interactiv. Pentru a rezolva aceste probleme, a fost inventată **comutarea de pachete**, așa cum a fost descrisă în cap. 1. Rețelele cu comutare de pachete fixează o limită superioară precisă pentru dimensiunea blocului, permițând pachetelor să fie păstrate în memoria principală a ruterului, în loc să fie salvate pe disc. Asigurându-se faptul că nici un utilizator nu va putea monopoliza o linie de transmisie mult timp (milisecunde), rețelele cu comutare de pachete au devenit adecvate pentru traficul interactiv. Un alt avantaj al comutării de pachete față de comutarea de mesaje este prezentat în fig. 2-39(b) și (c): primul pachet al unui mesaj multi-pachet poate fi transmis mai departe înainte ca cel de-al doilea pachet să fie complet recepționat, micșorând întârzierea și îmbunătățind productivitatea. Din aceste motive, rețelele de calculatoare folosesc, de obicei, comutarea de pachete, ocazional, comutarea de circuite, dar niciodată comutarea de mesaje.

Comutarea de circuite și comutarea de pachete diferă în multe privințe. Pentru început, comutarea de circuite necesită construirea unui circuit între transmițător și receptor înainte să înceapă comunicația. Comutarea de pachete nu are nevoie de pregătiri prealabile. Primul pachet poate fi trimis atunci când este disponibil.

Rezultatul unei conexiuni stabilite cu comutare de circuite este rezervarea lărgimii de bandă pe tot traseul de la transmițător la receptor. Toate pachetele urmează această cale. Faptul că toate pachetele urmează aceeași cale înseamnă că nu pot ajunge la destinație în altă ordine decât aceea în care au fost trimise. La comutarea cu pachete nu există o cale, deci pachete diferite pot urma căi diferite, în funcție de condițiile de rețea din momentul în care sunt trimise. Acestea pot ajunge în altă ordine decât cea inițială.

Comutarea cu pachete este mult mai tolerantă la erori decât comutarea cu circuite. De fapt, acesta este motivul pentru care a fost inventată. Dacă un comutator se defectează, toate circuitele care îl folosesc se termină și nu se mai poate face trafic pe acestea. În cazul comutării cu pachete, pachetele pot fi redirecționate astfel încât să poată ocoli comutatoarele defecte.

De asemenea, rezervarea unei căi oferă posibilitatea rezervării de lățime de bandă. Dacă lățimea de bandă e rezervată, atunci când un pachet ajunge la destinație el este transmis imediat mai departe. În cazul comutării de pachete, lățimea de bandă nu este rezervată, și deci pachetele ar putea aștepta până să fie trimise mai departe.

Prin rezervarea lățimii de bandă se asigură că nu poate să apară congestie când sosește un pachet (doar dacă apar mai multe pachete decât sunt așteptate). Pe de altă parte, când se încearcă stabilirea unui circuit, există posibilitatea unui eșec datorat congestiei. Deci, congestia poate interveni la momente diferite în cazul comutării de circuite (la stabilirea circuitului) și în cazul comutării de pachete (la transmisia pachetele).

Dacă un circuit a fost rezervat pentru un anumit utilizator și nu există trafic, lățimea de bandă a circuitului e irosită, și nu poate fi folosită pentru alt trafic. Comutarea de pachete nu irosește lățime de bandă, deci este mai eficientă dintr-o perspectivă de ansamblu. Înțelegerea acestui compromis este crucială pentru înțelegerea diferenței dintre comutarea de circuite și comutarea de pachete. Compromisul este între serviciul garantat cu resurse irosite și serviciul negarantat cu resurse neirosite.

Comutarea de pachete folosește transmisia de tip memorează-și-trimite. Un pachet e salvat în memoria unui ruter, apoi trimis către următorul ruter. În cazul comutării de circuite, biții curg continuu prin fir. În cazul comutării de pachete, transmisia memorează-și-trimite adaugă o întârziere.

O altă diferență este transparența completă a comutării de circuite. Transmițătorul și receptorul pot folosi orice viteză de transfer, orice format sau orice metodă de formare a cadrului. Compania de telecomunicații nu cunoaște aceste lucruri și nici nu este interesată de ele. În cazul comutării de pachete, compania de telecomunicație determină parametrii de bază. O analogie grosieră ar fi o comparație între șosea și calea ferată. În cazul celei dintâi, utilizatorul determină mărimea, viteza și natura vehiculului; în cazul celei de-a doua, acest lucru îl face societatea de cale ferată. Această transparență face posibilă coexistența vocii, a faxurilor și a datelor în sistemul telefonic.

O ultimă diferență între comutarea de circuite și comutarea de pachete se referă la algoritmul de taxare. La comutarea de circuite, acesta a fost bazat de la început pe distanță și timp. De obicei, pentru telefoanele mobile distanța nu contează, excepție făcând convorbirile internaționale, iar timpul are doar un rol minor (de exemplu un abonament cu 2000 de minute gratuite costă mai mult decât unul cu 1000 de minute gratuite, iar uneori convorbirile în timpul nopții și în weekend sunt mai ieftine decât în mod normal). La comutarea de pachete, timpul de conectare nu contează, dar volumul de trafic da. Pentru utilizatorii simpli, distribuitorii de Internet taxează o anumită sumă lunar pentru că e mai puțin de lucru pentru ei și mai ușor de înțeles de către clienți, dar rețelele de infrastructură taxează rețelele regionale pe baza volumului de trafic. Diferențele sunt prezentate în fig. 2-40.

Criteriu	Comutarea de circuite	Comutarea de pachete
Realizarea conectării	Necesară	Nu e necesară
Cale fizică dedicată	Da	Nu
Fiecare pachet urmează aceiași cale	Da	Nu
Pachetele ajung în ordine	Da	Nu
Defectarea unui comutator e fatală	Da	Nu
Banda de frecvență disponibilă	Fixă	Dinamică
Când poate să apară congestia	La momentul setării	La fiecare pachet
Banda de frecvență eventual risipită	Da	Nu
Transmisia memorează și transmite	Nu	Da
Transparență	Da	Nu
Taxarea	Pe minut	Pe pachet

Fig. 2-40. Comparație între rețelele cu comutare de circuite și cu comutare de pachete.

Datorită faptului că atât comutarea de circuite cât și comutarea de pachete sunt foarte importante, vom reveni la ele în curând și vom prezenta în detaliu diversele tehnologii folosite.

2.6 SISTEMUL DE TELEFONIE MOBILĂ

Sistemul tradițional de telefonie (chiar dacă uneori înseamnă fibră optică multi-megabit), nu va putea satisface un grup tot mai mare de utilizatori: oamenii în mișcare. Oamenii se așteaptă să efectueze convorbiri telefonice din avion, din mașină, din piscină sau în timp ce aleargă prin parc. Nu mai departe de câțiva ani se vor aștepta să trimită e-mail-uri și să navigheze pe Internet din toate aceste locuri și din altele. Prin urmare, există un interes extrem de mare în telefonia fără fir. În următoarele paragrafe vom studia în detaliu acest subiect.

Telefoanele fără fir există în două variante de bază: telefoanele fără fir și telefoanele mobile (uneori numite și **celulare**). **Telefoanele fără fir** sunt dispozitive constând dintr-o stație bază și un

receptor, vândute ca set pentru uzul casnic. Aceste dispozitive nu sunt folosite pentru rețele, deci nu le vom mai prezenta în continuare. În schimb ne vom concentra asupra telefoanelor mobile, care sunt folosite pentru comunicația de date și voce pe arii mari.

Telefoanele mobile sunt împărțite în trei generații, fiecare cu o tehnologie diferită:

1. Voce analogic.
2. Voce digital.
3. Voce și date (Internet, e-mail, etc.) digital.

Deși mare parte a discuției se va referi la tehnologia acestor sisteme, e interesant de văzut în ce fel politica și micile decizii de marketing pot avea un impact uriaș. Primul sistem mobil din SUA a fost inventat de AT&T și extins la nivelul întregii țări de FCC. Drept urmare, pe tot teritoriul SUA era un singur sistem (analogic), iar un telefon cumpărat în California funcționa și în New York. În contrast, când telefonia mobilă a ajuns în Europa, fiecare țară și-a inventat propriul sistem, rezultatul fiind un fiasco.

Europa a învățat din propriile greșeli și când s-a pus problema digitizării, guvernele s-au întâlnit și au standardizat un singur sistem (GSM), așa încât orice telefon mobil european să poată funcționa oriunde în Europa. Până atunci, SUA hotărâse că guvernul nu ar trebui să intervină în procesul de standardizare, deci a lăsat digitizarea pe mâna pieței. Această decizie a dus la telefoane diferite, fabricate de producători diferiți. Drept urmare, SUA are două sisteme digitale mari incompatibile (plus încă unul mai mic).

În ciuda unui avans inițial în SUA, deținerea și folosirea unui telefon mobil în Europa este mult mai mare ca în SUA. Crearea unui singur sistem pentru toată Europa este unul din motive, dar mai sunt și altele. Un al doilea domeniu unde Europa și SUA se diferențiază este problema atribuirii numerelor de telefon. În SUA, numerele de telefoane mobile sunt amestecate cu cele de telefoane normale (fixe). Deci, este imposibil pentru o persoană care sună , să zicem la (212) 234+5678 să știe dacă e un post telefonic fix (mai ieftin sau gratis) sau un telefon mobil (convorbire scumpă). Pentru a preveni enervarea oamenilor la folosirea telefoanelor, companiile telefonice au decis ca posesorul telefonului mobil să plătească apelurile recepționate. Drept urmare, mulți oameni au ezitat sa-și cumpere un telefon mobil de frică să nu ajungă cu o notă de plată foarte mare doar pentru că au fost sunați. În Europa, telefoanele mobile au un prefix special (analog numerelor cu 800 si 900) deci sunt foarte ușor de recunoscut. Prin urmare, regula că acela care sună plătește se aplică și telefoanelor mobile în Europa (cu excepția convorbirilor internaționale, unde costul este împărțit).

Un al treilea aspect care a avut un impact semnificativ este folosirea telefoanelor preplătite în Europa, (până la 75 % în unele zone). Acestea pot fi cumpărate din multe magazine, cu cât mai puține formalități, nu mai multe decât pentru cumpărarea unui radio. Plătești și pleci. Acestea sunt preîncărcate cu, de exemplu, 20 sau 50 Euro și pot fi reîncărcate (folosind un cod PIN secret) când suma se termină. În consecință, practic fiecare adolescent și mulți dintre copiii mici din Europa au telefoane mobile (de obicei preîncărcate), pentru ca părinții să îi poată găsi, fără să existe pericolul unei note de plată imense pentru telefonul copilului. Dacă telefonia mobilă e folosită doar ocazional, este practic gratuită, din moment ce nu există o taxă lunară sau taxare pentru apelurile recepționate.

2.6.1 Prima generație de telefoane mobile: Voce analogică

Am discutat suficient despre politica și aspectele de marketing ale telefoanelor mobile. Acum să ne uităm la tehnologie, pornind cu primele sisteme. Radiotelefoanele mobile au fost folosite spora-

dic, pentru comunicații maritime și militare, încă din timpul primelor decenii ale secolului XX. În 1946, primul sistem de telefoane pentru automobile a fost pus în funcțiune în St. Louis. Acest sistem folosea un singur emițător amplasat pe o clădire înaltă și avea un singur canal folosit atât pentru emisie cât și pentru recepție. Pentru a vorbi, un utilizator trebuia să apese un buton care activa emițătorul și dezactiva receptorul. Astfel de sisteme, cunoscute sub denumirea de **sisteme cu buton de emisie** au fost instalate în câteva orașe, spre sfârșitul anilor 1950. Radioul CB, taxiurile și mașinile de poliție folosesc deseori această tehnologie.

În 1960 a fost instalat **IMTS** (**Improved Mobile Telephone System**, rom: sistem îmbunătățit de telefonie mobilă). Și acesta folosește un emițător de mare putere (200 W), amplasat pe vârful unui deal, dar utilizează două frecvențe: una pentru emisie și una pentru recepție. Prin urmare, nu mai este nevoie de butonul de emisie. Deoarece toată comunicația dinspre telefoanele mobile se desfășoară pe un canal diferit de cel pe care telefoanele ascultă, utilizatorii telefoanelor mobile nu se mai pot auzi unii pe alții (spre deosebire de sistemul cu buton de emisie folosit la taxiuri).

IMTS suportă 23 de canale distribuite între 150 MHz și 450 MHz. Din cauza numărului mic de canale, utilizatorii trebuie să aștepte deseori perioade lungi de timp până când obțin tonul. De asemenea, datorită puterii mari a emițătorului aflat la înălțime, sistemele adiacente trebuie să se afle la câteva sute de kilometri distanță, pentru a se evita interferența. În concluzie, sistemul a fost impracticabil datorită posibilităților limitate.

Sistemul avansat de telefonie mobilă

Totul s-a schimbat odată cu **AMPS** (**Advanced Mobile Phone System**, rom: sistem avansat de telefonie mobilă), inventat de Bell Labs și instalat pentru prima dată în S.U.A. în 1982. Sistemul a fost folosit și în Anglia, unde purta denumirea TACS și în Japonia, unde se numea MCS-L1. Deși nu mai este la modă, îl vom studia mai în detaliu, deoarece multe din proprietățile sale fundamentale au fost moștenite de succesorul sau digital, D-AMPS, din considerente de compatibilitate.

În toate sistemele de telefonie mobilă, o regiune geografică e împărțită în **celule** și de aceea telefoanele sunt uneori numite telefoane celulare. În AMPS, o celula are de obicei 10 până la 20 km lățime; în sistemele digitale celulele sunt mai mici. Fiecare celulă folosește un set de frecvențe, nefolosit de nici unul dintre vecinii săi. Ideea de bază care conferă sistemelor celulare o capacitate semnificativ mai mare decât a tuturor sistemelor anterioare, constă în folosirea de celule relativ mici și refolosirea frecvențelor de transmisie în celule apropiate (dar nu adiacente). În timp ce într-un sistem IMTS de 100 km lățime poate exista un singur apel pe fiecare frecvență, un sistem AMPS poate avea 100 de celule de 10 km în aceeași regiune și este capabil să suporte 10 până la 15 apeluri pe fiecare frecvență, în celule separate de distanțe mari. Astfel, proiectarea celulară determină creșterea capacității sistemului cu cel puțin un ordin de mărime, cu atât mai mult cu cât celulele devin mai mici. Mai mult, celulele de dimensiuni reduse necesită puteri mici, ceea ce implică folosirea de transmițătoare mai ieftine și de dimensiuni mai mici. Telefoanele de mână emit 0,6W; emițătoarele de pe mașini au de obicei 3 W, valoarea maximă permisă de FCC.

Ideea refolosirii frecvențelor este ilustrată în fig. 2-41(a). În mod normal, celulele sunt aproximativ circulare, dar ele pot fi modelate mai ușor ca hexagoane. În fig. 2-41(a), celulele au toate aceeași dimensiune. Ele sunt grupate împreună în unități de 7 celule. Fiecare literă indică un grup de frecvențe. Trebuie remarcat că pentru fiecare set de frecvențe există o zonă tampon, de lățime aproximativ egală cu dublul mărimii unei celule, în care acea frecvența nu este refolosită, realizând astfel o delimitare mai bună și o interferență scăzută.

O problemă majoră o constituie găsirea locurilor înalte pentru instalarea antenelor stației de bază. Această problemă a determinat pe unii furnizori de servicii de telecomunicații să încheie contracte cu Biserica Romano-catolică pentru că aceasta dispune de un număr substanțial de potențiale locuri înalte pentru antene, toate aflate, în mod convenabil, sub o singură administrație.

Într-o zonă în care numărul de utilizatori s-a mărit atât de mult încât sistemul a devenit supraîncărcat, se reduce puterea, iar celulele supraîncărcate se divizează în **microcelule,** pentru a permite mai multe refolosiri de frecvențe, așa cum este arătat în fig. 2-41(b). Determinarea dimensiunii maxime a celulelor constituie o problemă complexă și este tratată în (Hac, 1995).

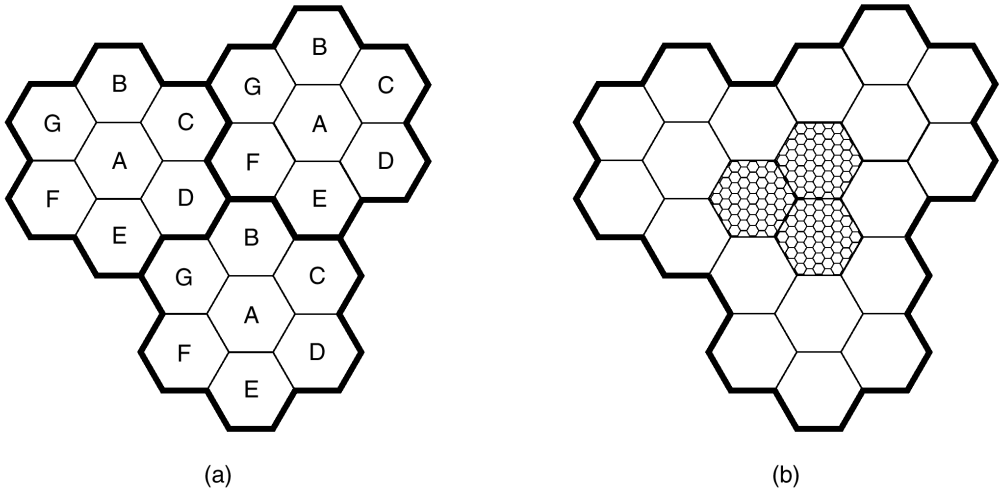


Fig. 2-41. (a) Frecvențele nu sunt refolosite în celule adiacente.
(b) Pentru a adăuga mai mulți utilizatori se pot folosi celule mai mici.

În centrul fiecărei celule se află o stație de bază către care transmit toate telefoanele din celulă. Stația de bază cuprinde un calculator și un emițător/receptor conectat la o antenă. Într-un sistem de dimensiuni reduse, toate stațiile de bază sunt conectate la un singur dispozitiv, denumit **MTSO** (**Mobile Telephone Switching Office**, rom: oficiu de comutare pentru telefonie mobilă) sau **MSC** (**Mobile Switching Center**, rom: centru de comutare mobil). Într-un sistem de dimensiuni mari, pot fi necesare mai multe MTSO-uri, toate acestea conectându-se la un al doilea nivel MTSO și așa mai departe. MTSO-urile sunt în esență oficii finale – ca și în sistemul telefonic – și sunt de fapt conectate la cel puțin un oficiu final din sistemul telefonic. MTSO-urile comunică între ele, cu stațiile de bază și cu PSTN-ul folosind o rețea cu comutare de pachete.

În orice moment, orice telefon mobil se află, în mod logic, într-o anumită celulă și sub controlul stației de bază a celulei respective. Când un telefon mobil părăsește o celulă, stația sa de bază sesizează o scădere a semnalului dinspre telefon și întreabă toate stațiile de bază înconjurătoare cât de puternic este semnalul pe care îl recepționează de la respectivul telefon. Stația de bază transferă apoi proprietatea asupra telefonului către celula care recepționează cel mai puternic semnal, aceasta fiind și celula în care se află acum telefonul. Telefonul este informat despre noul său șef, iar dacă un apel este în derulare în acel moment, telefonul va fi rugat să comute pe un canal nou (deoarece canalul vechi nu este refolosit în nici o celulă adiacentă). Acest proces poartă denumirea de **timp mort** și durează aproximativ 300 ms. Atribuirea canalului se face de către MTSO, care este centrul nervos al sistemului. Stațiile de bază sunt, de fapt, doar niște radio-relee.

Timpii morți pot fi eliminați în două feluri. Într-un **timp mort soft** telefonul e luat în primire de noua stație de bază înainte ca stația veche să îl cedeze. Astfel, nu apare nici o discontinuitate. Partea proastă a acestei variante este că telefonul trebuie să fie capabil să se seteze pe două frecvențe în același timp (cea veche și cea nouă). Nici telefoanele de primă generație, nici cele de generația a doua nu pot face acest lucru.

Într-un **timp mort hard**, vechea stație de bază deconectează telefonul înainte ca stația nouă să îl preia. Dacă noua stație de bază nu poate să preia telefonul (de ex. pentru că nu există nici o frecvență disponibilă), convorbirea se termină brusc. Utilizatorii constată acest lucru, dar câteodată situația este inevitabilă din cauza modului în care sunt proiectate telefoanele actuale.

Canale

Sistemul AMPS utilizează 832 canale full-duplex, fiecare constând dintr-o pereche de canale simplex. Există astfel 832 canale simplex pentru transmisie de la 824 la 849 MHz și 832 canale simplex pentru recepție de la 869 la 894 MHz. Fiecare din aceste canale simplex are o lățime de 30 KHz. Din această cauză AMPS folosește FDM pentru a separa canalele.

În banda de 800 MHz, undele radio au aproximativ 40 cm lungime și se propagă în linie dreaptă. Ele sunt absorbite de copaci și plante și sunt deviate de pământ și clădiri. Este posibil ca un semnal emis de un telefon mobil să ajungă la stația de bază pe calea directă, dar tot la fel de bine poate să ajungă puțin mai târziu, după ce este deviat de pământ sau clădiri. Aceasta poate să conducă la un efect de ecou sau la distorsionarea semnalului (atenuare multi-căi). Uneori este posibil să se audă chiar și o convorbire îndepărtată care a suferit mai multe deviații. Cele 832 de canale se împart în patru categorii:

1. Comandă (baza către mobil) pentru gestionarea sistemului.
2. Semnalizare (baza către mobil) pentru a anunța utilizatorii de telefoane mobile că sunt apelați.
3. Acces (bidirecțional) pentru stabilirea apelului și alocarea canalului.
4. Date (bidirecțional) pentru voce, fax sau date.

Pentru comenzi sunt rezervate douăzeci și unu de canale și acestea sunt fixate în fiecare telefon într-un PROM. Deoarece aceleași frecvențe nu pot fi refolosite în celule învecinate, numărul real de canale de voce disponibile pe celulă este mult mai mic decât 832, de regulă 45.

Gestiunea apelului

Fiecare telefon mobil din AMPS are un număr serial pe 32 biți și un număr de telefon de 10 cifre în PROM-ul propriu. Numărul de telefon este format dintr-un cod al zonei de 3 cifre pe 10 biți și un număr de abonat de 7 cifre pe 24 de biți. Atunci când este activat, un telefon scanează o listă preprogramată cu 21 canale de comandă, pentru a descoperi semnalul cel mai puternic.

Apoi telefonul difuzează propriul număr serial de 32 de biți și numărul de telefon de 34 de biți. Ca orice altă informație de comandă din AMPS, acest pachet este transmis în formă digitală, de mai multe ori și cu un cod corector de erori, deși canalele de voce sunt analogice.

Atunci când stația de bază aude anunțul, sesizează MTSO-ul care înregistrează existența noului său client și informează de asemenea MTSO-ul clientului asupra poziției sale curente. În timpul unei funcționări normale, telefonul mobil se reînregistrează la fiecare aproximativ 15 minute.

Pentru a face un apel, un utilizator de telefon mobil activează telefonul, introduce de la taste numărul de apelat și apasă butonul SEND. Telefonul transmite apoi numărul de apelat și identitatea proprie pe canalul de acces. Dacă pe canalul de acces apare o coliziune, se încearcă din nou mai târziu. Atunci când primește o cerere, stația de bază informează MTSO-ul. Dacă apelantul este un client al companiei MTSO (sau unul din parteneri), MTSO-ul caută un canal liber pentru apel. Dacă

se găsește unul, numărul canalului este transmis înapoi pe canalul de comandă. Telefonul mobil comută apoi automat pe canalul de voce selectat și așteaptă până când partea apelată ridică telefonul.

Apelurile primite acționează diferit. La început, toate telefoanele libere ascultă continuu canalul de semnalizare (paging) pentru a detecta mesajele adresate lor. Atunci când se face apel către un telefon mobil (fie de la un telefon fix, fie de la un alt telefon mobil), se transmite un pachet către MTSO-ul apelatului pentru a descoperi unde se află acesta. Se transmite apoi un pachet către stația de bază din celula sa curentă, care transmite apoi pe canalul de semnalizare (paging) un mesaj de difuzare de forma următoare: „Unitatea 14, ești acolo?". Telefonul apelat răspunde apoi cu „Da" pe canalul de acces. Baza spune apoi ceva de genul „Unitatea 14, ai un apel pe canalul 3". În acest moment, telefonul apelat comută pe canalul 3 și începe să sune (sau cântă o melodie pe care proprietarul a primit-o cadou de ziua lui).

2.6.2 A doua generație de telefoane mobile: Voce digitală

Prima generație de sisteme celulare a fost analogică. Cea de-a doua generație este digitală. Așa cum nu a existat o standardizare la prima generație, nu a existat nici la a doua. Patru sisteme se folosesc în prezent. D-AMPS, GSM, CDMA și PDC. Mai jos le vom discuta pe primele trei. PDC e folosit doar în Japonia și este practic D-APMS modificat pentru a respecta compatibilitatea cu sistemul analogic de primă generație din Japonia. Numele de **PCS** (**Personal Communications Services**, rom: serviciu de comunicații personale) este folosit câteodată în literatura de marketing pentru a indica un sistem de generația a doua (adică digital). Inițial se referea la un telefon mobil care folosea banda de 1900 MHz, dar distincția nu se mai face acum.

D-AMPS – Sistem digital avansat de telefonie mobilă

A doua generație a sistemelor AMPS este **D-AMPS** (**The Digital Advanced Mobile Phone System**, rom: sistem digital avansat de telefonie mobilă) și este complet digital. Este descris în Standardul Internațional IS-54 și în succesorul acestuia, IS-136. D-AMPS a fost proiectat cu atenție pentru a coexista cu AMPS, așa încât ambele generații de telefoane mobile , și cele de primă generație și cele de a doua, să poată opera simultan în aceeași celulă. D-AMPS folosește aceleași canale de 30KHz ca și AMPS și aceleași frecvențe, astfel încât un canal poate fi analogic și cele adiacente digitale. În funcție de ponderile tipurilor telefoanelor dintr-o celulă, MTSO-ul celulei determină care canale sunt analogice și care sunt digitale și poate modifica dinamic tipul canalului, după cum se schimbă ponderile telefoanelor în celulă.

Când D-AMPS a fost introdus ca serviciu, s-a adăugat o nouă bandă de frecvență disponibilă pentru a gestiona încărcarea care se aștepta să crească. Canalele ascendente (de transmisie) erau în domeniul de frecvențe 1880-1910 MHz, iar cele descendente (de recepție) corespunzătoare erau în domeniul 1930-1990 MHz, organizate tot în perechi, ca și la AMPS. În această bandă, undele au 16 cm lungime, așa că antena standard ¼ din lungimea de undă este de numai 4 cm, rezultând astfel telefoane mai mici. Oricum, multe telefoane D-AMPS pot folosi atât banda de 850 MHz cât și pe cea de 1900 MHz, pentru a avea un domeniu mai mare de canale disponibile.

Pe un telefon mobil D-AMPS, semnalul de voce captat de microfon este digitizat și compresat folosind un model mai complicat decât varianta cu modulație delta și codificare predictivă studiată mai devreme. Compresia ia în considerare proprietăți detaliate ale vocii umane pentru a transforma lărgimea de bandă standard de la codarea PCM (56 Kbps) la 8 Kbps sau mai puțin. Compresia este realizată de un circuit numit **vocoder** (Bellamy,2000). Compresia este făcută în telefon, și nu în stația

de bază sau la oficiul final, pentru a se reduce numărul de biți trimiși prin aer. În cazul telefoniei fixe, nu există nici un beneficiu dacă se face compresia în telefon, pentru că reducerea traficului pe bucla locală nu mărește deloc capacitatea sistemului.

La telefonia mobilă se câștigă atât de mult prin digitizarea și compresia în telefon, încât la D-AMPS trei utilizatori pot împărți o pereche de frecvențe folosind multiplexarea prin divizare în timp. Fiecare pereche de frecvențe suportă 25 cadre/sec, adică 40 ms pentru fiecare cadru. Fiecare cadru este împărțit în șase intervale de timp de câte 6,67 ms, după cum se arată în fig. 2-42(a) pentru cea mai joasă pereche de frecvențe.

Fiecare cadru cuprinde trei utilizatori, care stau la rând pentru a transmite și a recepționa. În timpul intervalului 1 din fig. 2-42 (a), de exemplu, utilizatorul 1 poate transmite către stația de bază și utilizatorul 3 recepționează de la stația de bază. Fiecare interval cuprinde 324 de biți, dintre care 64 sunt folosiți pentru timpii de gardă, sincronizare și comandă, lăsând 260 de biți pentru informația utilă utilizatorului. Dintre biții de informație utilă, 101 sunt folosiți pentru corectarea erorilor datorate mediului aerian perturbat și în final rămân doar 159 de biți pentru semnalul de voce comprimat. Având 50 de intervale/sec, lățimea de bandă disponibilă pentru semnalul de voce comprimat este puțin sub 8 Kbps, adică 1/7 din lărgimea de bandă standard de la PCM.

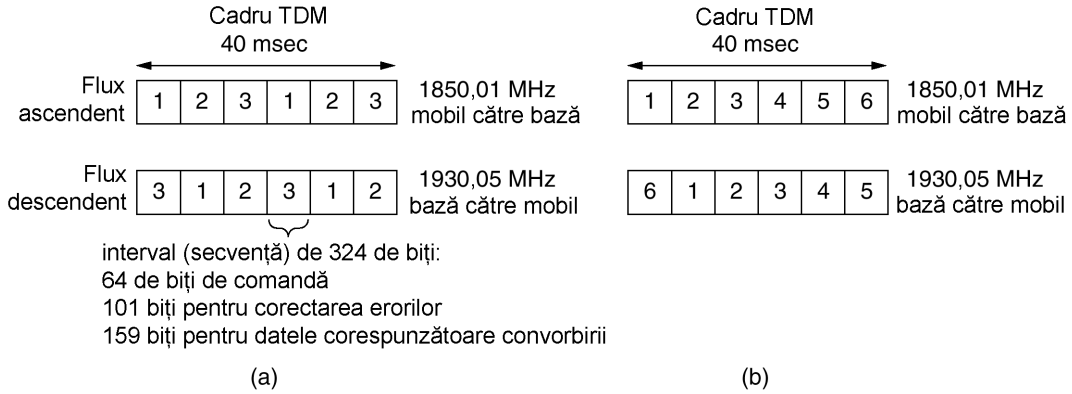

Cadru TDM
40 msec
Flux ascendent
1 2 3 1 2 3
1850,01 MHz mobil către bază
Flux descendent
3 1 2 3 1 2
1930,05 MHz bază către mobil
interval (secvență) de 324 de biți:
64 de biți de comandă
101 biți pentru corectarea erorilor
159 biți pentru datele corespunzătoare convorbirii
(a)
Cadru TDM
40 msec
1 2 3 4 5 6
1850,01 MHz mobil către bază
6 1 2 3 4 5
1930,05 MHz bază către mobil
(b)

Fig. 2-42. (a) Un canal S-AMPS cu trei utilizatori. (b) un canal D-AMPS cu șase utilizatori

Folosind algoritmi de compresie mai buni este posibil să se reducă datele de voce la 4 Kbps, caz în care șase utilizatori pot fi cuprinși într-un cadru, după cum se arată în fig. 2-42(b). Din punctul de vedere al operatorului, posibilitatea de a înghesui de trei sau de șase ori mai mulți utilizatori D-AMPS în același spectru dedicat unui singur utilizator de AMPS este un câștig uriaș și așa se explică popularitatea PCS-ului. Bineînțeles, calitatea vorbirii la 4 Kbps nu se compară cu cea atinsă la 56 Kbps, dar puțini operatori PCS își fac reclame cu referire la calitatea sunetului. De asemenea, ar trebui să fie clar că, pentru date, un canal de 8 Kbps nu este nici măcar la fel de bun ca un antic modem de 9600 bps.

Structura de comandă a D-AMPS este destul de complicată. Descriere sumară: grupuri de 16 cadre formează un supercadru, cu informații de comandă specifice prezente în fiecare supercadru de un număr limitat de ori. Șase canale principale de comandă sunt folosite pentru configurarea sistemului, comanda în timp real sau offline, semnalizare, răspunsul la cererea de acces și mesajele scurte. În principiu, funcționează ca un AMPS. Când un mobil este deschis, el contactează stația de bază, pentru a-și anunța apariția, și apoi ascultă pe un canal de comandă eventualele apeluri. Când recepționează un mobil nou, MTSO informează baza utilizatorului despre locul unde se află acesta, pentru a se putea ruta corect convorbirile.

O diferență între AMPS și D-AMPS este modul în care se tratează timpii morți. În AMPS, MTSO se descurcă singur, fără nici un ajutor de la dispozitivul mobil. După cum se poate vedea în fig. 2-42, în D-AMPS, 1/3 din timp telefonul nici nu trimite nici nu primește. Ele folosesc acest timp pentru a măsura calitatea liniei. Când descoperă că semnalul începe să slăbească, anunță MTSO, care apoi poate întrerupe conexiunea, moment în care mobilul poate încerca să se conecteze la o altă stație de bază cu un semnal mai puternic. La fel ca la AMPS, acest timp mort durează totuși 300msec. Această tehnică se numește **MAHO** (**Mobile Assisted HandOff**, rom: timpi morți asistați de mobil).

GSM – Sistemul global pentru comunicații mobile

D-AMPS este larg folosit în SUA și (într-o formă modificată) în Japonia. Practic, în rest, peste tot în lume se folosește un sistem numit **GSM** (**Global System for Mobile Communications**, rom: sistemul global pentru comunicații mobile), care a început să fie folosit și în SUA la o scară limitată. La o privire generală, GSM este similar D-AMPS-ului. Amândouă sunt sisteme celulare. În ambele sisteme se folosește multiplexarea prin divizarea de frecvență, fiecare mobil transmițând pe o frecvență și recepționând pe una mai ridicată (cu 80 MHz mai mare pentru D-AMPS și cu 55 MHz mai mare pentru GSM). De asemenea, în ambele sisteme se folosește multiplexarea cu divizare în timp pentru a împărți o singură pereche de frecvențe în intervale de timp folosite în comun de mai multe dispozitive mobile. Oricum, canalele GSM sunt mult mai largi decât canalele AMPS (200kHz față de 30 KHz) și servesc doar câțiva utilizatori în plus (8 față de 3), permițând GSM-ului o rată de transmisie per utilizator mult mai mare decât la D-AMPS.

Mai jos vom discuta pe scurt unele dintre proprietățile principale ale GSM. Oricum, standardul GSM are peste 5000 [sic] de pagini. O mare parte din acest material se referă la aspecte inginerești ale sistemului, în special la proiectarea receptoarelor pentru a fi capabile să gestioneze propagarea semnalelor pe mai multe căi, și sincronizarea transmițătoarelor și receptoarelor. Nici unul dintre acestea nu va fi menționat în continuare.

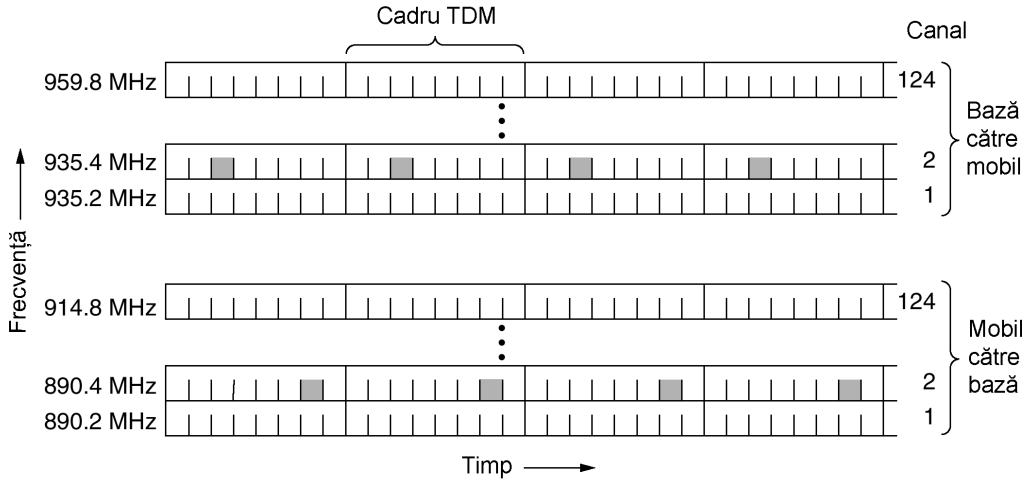


Fig. 2-43. GSM folosește 124 canale de frecvență, fiecare dintre ele folosind un sistem TDM cu opt intervale

Fiecare bandă de frecvență are 200 KHz, după cum se arată în fig. 2-43. Un sistem GSM are 124 de perechi de canale simplex. Fiecare canal simplex, are 200 KHz și suportă opt conexiuni diferite, folosind multiplexarea prin divizarea în timp. Fiecărei stații active la momentul respectiv i se atribuie un interval de timp dintr-o pereche de canale. Teoretic, pot fi suportate 992 de canale în fiecare celulă, dar multe dintre ele nu sunt disponibile, pentru a se evita conflictele de frecvență, cu celulele vecine. În fig. 2-43, toate cele opt intervale de timp marcate aparțin aceleiași conexiuni, câte patru dintre ele în fiecare direcție. Transmisia și recepția nu se fac în același interval de timp pentru că aparatele GSM nu pot trimite și primi în același timp și durează până se face comutarea între emisie și recepție. Dacă stația mobilă asociată frecvenței 890.4/935.4 MHz și intervalului de timp 2 ar vrea să transmită la stația de bază, trebuie să folosească acele patru intervale de timp inferioare (și pe cele care le urmează cronologic), punând informație în fiecare interval până când toată informația este transmisă.

Intervalele TDM prezentate în fig. 2-43 fac parte dintr-o ierarhie complexă de cadre. Fiecare interval TDM are o structură specifică, iar grupurile de intervale TDM formează multicadre, și acestea având o structură specifică. O variantă simplificată a acestei ierarhii este prezentată în fig. 2-44. Aici putem vedea că fiecare interval TDM constă dintr-un cadru de 148 de biți care ocupă canalul pentru 577 μs (incluzând și un timp de gardă de 30 μs după fiecare interval). Fiecare cadru de date începe și se termină cu trei biți de 0, pentru aliniere. De asemenea conține două câmpuri de *informație* de 57 de biți, fiecare având un bit de comandă care spune dacă următorul cadru va fi de date sau de voce. Între câmpurile de *informație* se află un câmp de 26 de biți de *sincronizare* (antrenare) care este folosit de receptor pentru sincronizarea la extremitățile cadrului de la transmițător.

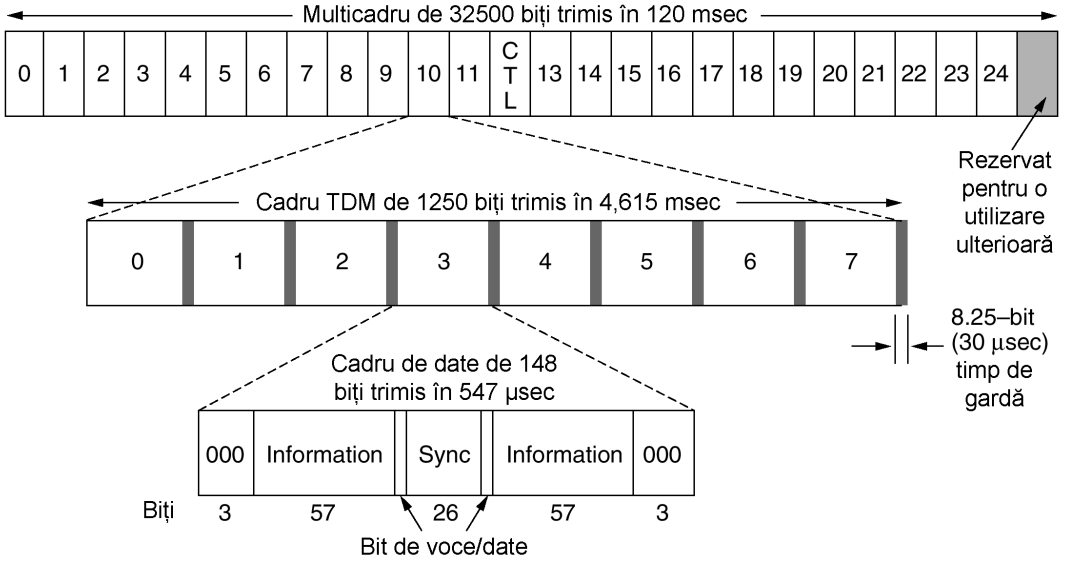

Multicadru de 32500 biți trimis în 120 msec
0 1 2 3 4 5 6 7 8 9 10 11 CTL 13 14 15 16 17 18 19 20 21 22 23 24
Rezervat pentru o utilizare ulterioară
Cadru TDM de 1250 biți trimis în 4,615 msec
0 1 2 3 4 5 6 7
8.25–bit (30 μsec) timp de gardă
Cadru de date de 148 biți trimis în 547 μsec
000 Information Sync Information 000
Biți 3 57 26 57 3
Bit de voce/date

Fig. 2-44. Un fragment din structura de cadre GSM

Un cadru de date e transmis în 547 μsec, dar un transmițător are voie să trimită câte un cadru de date la fiecare 4,615 msec, deoarece este pe același canal cu alte șapte stații. Rata totală a fiecărui canal este de 270.833 bps, împărțită între opt utilizatori. Aceasta înseamnă 33.854 Kbps în total, mai mult decât dublu față de D-AMPS, 324 de biți de 50 de ori pe secundă, adică 16,2 Kbps. Oricum, ca și la APMS, informația suplimentară consumă o mare parte din lărgimea de bandă, lăsând 24,7 Kbps fiecărui utilizator pentru informația utilă, înainte de corectarea erorilor. După corectarea erorilor,

13 Kbps rămân pentru voce, rezultând o calitate mult mai bună decât la D-AMPS (dar folosind mai multă lățime de bandă).

După cum se poate vedea în fig. 2-44, opt cadre de date formează un cadru TDM și 26 cadre TDM formează un multicadru de 120 msec. Dintre cele 26 de cadre TDM dintr-un multicadru, cadrul 12 e folosit pentru comandă, iar cadrul 25 e rezervat pentru o utilizare ulterioară, deci numai 24 sunt disponibile efectiv pentru traficul utilizatorilor.

Oricum, pe lângă multicadrul de 26 de intervale din fig. 2-44, se folosește un multicadru de 51 de intervale (nu este desenat). Unele dintre aceste intervale sunt folosite pentru a menține câteva canale de comandă folosite pentru gestiunea sistemului. **Canalul de difuzare a comenzii** este un flux continuu de ieșire de la stația de bază, care conține elemente pentru identificarea stației și starea canalului. Toate stațiile mobile își monitorizează puterea semnalului, pentru a constata dacă s-au mutat într-o celulă nouă.

Canalul dedicat de comandă este folosit pentru actualizarea locației, înregistrarea și setările convorbirii. În particular, fiecare stație de bază reține o bază de date a stațiilor mobile care se află sub jurisdicția sa. Informațiile necesare pentru a menține această bază de date se trimit pe canalul dedicat de comandă.

În sfârșit, există canalul comun de comandă, care este împărțit în trei subcanale logice. Primul dintre ele este canalul de semnalizare, pe care stațiile de bază îl folosesc pentru a anunța apelurile primite. Fiecare mobil monitorizează continuu acest canal pentru a vedea dacă sunt convorbiri la care trebuie să răspundă. Al doilea este canalul de acces aleator, care le permite utilizatorilor să ceară un loc pe canalul dedicat de comandă. Dacă două cereri intră în coliziune, ele sunt bruiate și ambele trebuie să încerce mai târziu. Folosind canalul dedicat de comandă, stațiile pot inițializa un apel. Intervalul primit este anunțat prin cel de al treilea subcanal, **canalul de acces permis**.

CDMA – Acces multiplu cu divizarea codului

D-AMPS și GSM sunt sisteme convenționale. Amândouă folosesc FDM și TDM pentru a împărți spectrul în canale și canalele în intervale de timp. Mai există totuși un al treilea tip, **CDMA** (**Code Division Multiple Acces**, rom: acces multiplu cu divizarea codului) care funcționează complet diferit. Când CDMA a fost propus prima dată, industria i-a răspuns cu aceeași atitudine cu care regina Isabella i-a răspuns lui Columb când acesta i-a propus să ajungă în India navigând în direcția opusă. Totuși, prin insistența unei singure companii, Qualcomm, CDMA s-a maturizat într-atât încât nu este doar acceptat, ci este acum văzut ca fiind cea mai bună soluție tehnică disponibilă și de bază pentru sistemele mobile de generația a treia. De asemenea, în SUA se folosește pe scară largă în sistemele de generația a doua, concurând cu D-AMPS. Spre exemplu, Sprint PCS folosește CDMA, iar AT&T Wireless folosește D-AMPS. CDMA este descris în standardul internațional IS-95 și uneori este referit cu acest nume. Numele de marcă **cdmaOne** este de asemenea folosit.

CDMA e complet diferit de AMPS, D-AMPS și GSM. În loc să se împartă domeniul de frecvențe disponibil în câteva sute de canale mici, CDMA permite fiecărei stații să transmită tot timpul în tot spectrul. Mai multe transmisii simultane sunt separate folosind teoria codării. CDMA renunță la ideea că toate cadrele care intră în coliziune sunt total bruiate. În schimb, mai multe semnale recepționate vor fi sumate liniar.

Înainte de a prezenta algoritmul, să considerăm analogia cu o sală de așteptare a unui aeroport cu mai multe perechi de oameni discutând. TDM ar însemna că toți oamenii sunt în centrul încăperii, dar vorbesc pe rând. FDM ar însemna că fiecare om este într-un colț, fiecare vorbește în același timp cu ceilalți, dar independent de ei. CDMA ar însemna că fiecare persoană este în centrul încă-

perii, toți vorbesc în același timp, dar fiecare pereche de vorbitori conversează într-o limbă diferită. Cei ce vorbesc în franceză ascultă doar ceea ce este în franceză, ignorând orice nu este în limba franceză. Astfel, cheia CDMA este posibilitatea de a extrage semnalul dorit și de a respinge restul de semnale. O descriere simplificată a CDMA este prezentată mai jos.

În CDMA durata fiecărui bit este divizată în *m* intervale scurte numite **felii** (eng.: **chips**). În mod normal sunt 64 sau 128 de felii pe bit, dar în exemplul de mai jos vom folosi doar opt pentru simplificare. Fiecare stație are un cod unic pe *m biți* numit **secvență de felii** (**chip sequence**). Pentru a transmite un bit 1, o stație trimite propria secvență de felii. Pentru a transmite un bit 0, trimite complementul față de unu al secvenței sale de felii. Nici un alt șablon nu este permis. Deci, pentru $m=8$, dacă stației A îi este asociată secvența 00011011, ea trimite un bit 1 trimițând 0001011 și un bit 0 trimițând 11100100.

Creșterea cantității de informație de trimis de la *b* biți/sec la *mb* felii/sec se poate face doar dacă lărgimea de bandă disponibilă crește de *m* ori, făcând din CDMA o formă de comunicație cu spectru larg, (presupunând că nu se fac schimbări în tehnicile de modulație sau codificare). Dacă avem o bandă de 1MHz pentru 100 de stații, folosind FDM fiecare ar avea 10 KHz și ar trimite 10Kbps (presupunând un bit pe Hz). Cu CDMA, fiecare stație folosește toată banda de 1 MHz, deci rata feliilor este de 1 Mega pe secundă. Cu mai puțin de 100 de felii pe bit, lărgimea de bandă efectivă pentru fiecare stație este mai mare la CDMA decât la FDM, iar problema alocării canalului este de asemenea rezolvată.

A: 0 0 0 1 1 0 1 1	A: (–1 –1 –1 +1 +1 –1 +1 +1)
B: 0 0 1 0 1 1 1 0	B: (–1 –1 +1 –1 +1 +1 +1 –1)
C: 0 1 0 1 1 1 0 0	C: (–1 +1 –1 +1 +1 +1 –1 –1)
D: 0 1 0 0 0 0 1 0	D: (–1 +1 –1 –1 –1 –1 +1 –1)
(a)	(b)

Șase exemple:

– – 1 –	**C**	S_1 = (–1 +1 –1 +1 +1 +1 –1 –1)
– 1 1 –	**B** + $\overline{\mathbf{C}}$	S_2 = (–2 0 0 0 +2 +2 0 –2)
1 0 – –	**A** + $\overline{\mathbf{B}}$	S_3 = (0 0 –2 +2 0 –2 0 +2)
1 0 1 –	**A** + **B** + **C**	S_4 = (–1 +1 –3 +3 +1 –1 –1 +1)
1 1 1 1	**A** + **B** + **C** + **D**	S_5 = (–4 0 –2 0 +2 0 +2 –2)
1 1 0 1	**A** + **B** + $\overline{\mathbf{C}}$ + **D**	S_6 = (–2 –2 0 –2 0 –2 +4 0)

(c)

$S_1 \bullet C = (1 +1 +1 +1 +1 +1 +1 +1)/8 = 1$
$S_2 \bullet C = (2 +0 +0 +0 +2 +2 +0 +2)/8 = 1$
$S_3 \bullet C = (0 +0 +2 +2 +0 –2 +0 –2)/8 = 0$
$S_4 \bullet C = (1 +1 +3 +3 +1 –1 +1 –1)/8 = 1$
$S_5 \bullet C = (4 +0 +2 +0 +2 +0 –2 +2)/8 = 1$
$S_6 \bullet C = (2 –2 +0 –2 +0 –2 –4 +0)/8 = –1$

(d)

Fig.2-45. Secvențele de felii binare pentru patru stații. (b) Secvențele de felii bipolare. (c) Șase exemple de transmisii. (d) Recuperarea semnalului stației C

Din motive pedagogice, este mai convenabilă folosirea unei notații bipolare, cu 0 binar reprezentat ca -1 și 1 binar reprezentat ca +1. Vom scrie secvențele de felii în paranteze, deci un bit 1 pentru stația A devine (-1 -1 -1 +1 +1 -1 +1 +1). În fig. 2-45(a) sunt prezentate secvențele de felii binare asociate celor patru stații luate ca exemplu. În fig. 2-45(b) ele sunt prezentate în notația noastră bipolară.

Fiecare stație are propria sa secvență de felii. Vom folosi simbolul **S** pentru a indica vectorul de *m* felii pentru stația S, și $\overline{S}$ pentru negația sa. Toate secvențele de felii sunt **ortogonale pe perechi**, prin aceasta înțelegând că produsul scalar normat al oricăror două secvențe distincte de aschii **S** și **T** (notat ca **S•T**) este 0. Se cunoaște cum se generează asemenea secvențe de felii ortogonale, folosind metoda **codurilor Walsh**. În termeni matematici, ortogonalitatea secvențelor de felii poate fi exprimată după cum urmează:

$$\mathbf{S} \bullet \mathbf{T} \equiv \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0 \tag{2-4}$$

Altfel spus, toate perechile care se pot forma sunt diferite între ele. Această proprietate de ortogonalitate se va dovedi crucială mai târziu. De notat că dacă **S•T**=0 atunci și $\mathbf{S} \bullet \overline{\mathbf{T}} = 0$. Produsul scalar normat al oricărei secvențe de felii cu ea însăși este 1:

$$\mathbf{S} \bullet \mathbf{S} = \frac{1}{m} \sum_{i=1}^{m} S_i S_i = \frac{1}{m} \sum_{i=1}^{m} S_i^{\,2} = \frac{1}{m} \sum_{i=1}^{m} (\pm 1)^2 = 1$$

Acest lucru se întâmplă deoarece fiecare din cei *m* termeni ai produsului scalar este 1 deci suma este *m*. De asemenea se observă că $\mathbf{S} \bullet \overline{\mathbf{S}} = -1$.

În timpul fiecărui interval de bit, o stație poate să transmită un 1 emițând propria secvență de felii, sau poate transmite un 0 emițând complementul secvenței sale de felii, sau poate să nu transmită nimic. Pentru moment, vom presupune că toate stațiile sunt sincronizate în timp, deci toate secvențele de felii încep în același moment.

Când două sau mai multe stații transmit simultan, semnalele lor bipolare se adună liniar. De exemplu, dacă într-un interval de felie trei stații emit +1 și una -1, rezultatul este +2. Putem privi aceasta ca pe o adunare de tensiuni: trei stații emit +1 volt și o stație emite -1volt, rezultând 2 volți.

În fig. 2-45(c) avem șase exemple de una sau mai multe stații transmițând simultan. În primul exemplu, C transmite un bit de 1, deci avem doar secvența de felii a lui C. În al doilea exemplu, atât B cât și C transmit biți de 1, deci vom obține suma secvențelor lor de felii bipolare, și anume:

(-1 -1 +1 -1 +1 +1 +1 -1) + (-1 +1 -1 +1 +1 +1 -1 -1) = (-2 0 0 0 +2 +2 0 -2).

În cel de al treilea exemplu, stația A emite un 1 iar stația B emite un 0. Celelalte tac. În al patrulea exemplu, A și C emit câte un bit 1, în timp ce B emite un 0. În al cincilea exemplu, toate cele 4 stații emit câte un bit 1. În final, în ultimul exemplu, A, B și D emit câte un bit 1, în timp ce C emite un bit 0. Să notăm că fiecare dintre cele șase secvențe, S_1 – S_6, date în fig. 2-45(c), reprezintă durata unui singur bit.

Pentru a reface șirul de biți al unei stații individuale, receptorul trebuie să cunoască dinainte secvența de felii a stației. El face recuperarea calculând produsul scalar normat între secvența de felii recepționată (suma liniară a tuturor stațiilor care au transmis) și secvența de felii a stației al cărei șir de biți încearcă să-l refacă. Dacă secvența de felii recepționată e **S** și receptorul încearcă să asculte de la o stație a cărei secvență de felii e **C**, atunci va calcula doar produsul scalar normat **S•C**.

Pentru a vedea de ce se întâmplă așa, imaginați-vă două stații A și C, ambele transmițând un bit 1 în același timp în care B transmite un bit 0. receptorul primește suma $\mathbf{S=A+\overline{B}+C}$ și calculează:

$$\mathbf{S \cdot C = (A+\overline{B}+C) \cdot C = A \cdot C + \overline{B} \cdot C + C \cdot C = 0+0+1=1}$$

Primii doi termeni dispar, deoarece toate perechile de secvențe de felii au fost alese cu grijă pentru a fi ortogonale, ca în ecuația (2-4). Acum ar trebui să fie clar de ce această proprietate trebuie impusă secvențelor de felii.

Un alt mod de a gândi relativ la această situație este de a ne imagina toate cele trei secvențe de felii sosite separat, în loc să fie însumate. Atunci receptorul ar calcula produsul scalar cu fiecare în parte și ar aduna rezultatele. Datorită proprietății de ortogonalitate, toate produsele scalare în afară de $\mathbf{C \cdot C}$ ar fi 0. Adunându-le și apoi făcând produsul lor scalar, este de fapt același lucru cu calculul produselor scalare, și adunarea acestora.

Pentru a concretiza procesul de decodificare, să considerăm din nou cele șase exemple din fig. 2-45(c), ilustrat din nou în 2.45(d). Să presupunem că receptorul e interesat de extragerea bitului trimis de stația C din fiecare din cele șase sume $S_1 - S_6$. Se calculează acest bit prin însumarea perechilor de produse între vectorul **S** recepționat și vectorul **C** din figura2-45(b), luând apoi 1/8 din rezultat (pentru că în acest caz $m=8$). Așa cum am arătat, de fiecare dată este decodificat bitul corect. Este ca și cum s-ar fi vorbit franceza.

Într-un sistem CDMA ideal, fără zgomote, capacitatea (adică numărul de stații) poate fi mărită oricât de mult, așa cum și capacitatea unui canal Nyquist fără zgomote poate fi mărită oricât de mult, prin utilizarea unui număr tot mai mare de biți pe eșantion. În practică, limitările fizice reduc considerabil capacitatea. La început, am presupus că toate feliile sunt sincronizate în timp. În realitate, aceasta este imposibil de realizat. Ceea ce se poate face este ca emițătorul și receptorul să se sincronizeze prin expedierea de către emițător a unei secvențe de felii cunoscute, suficient de lungă pentru ca receptorul să o poată localiza. Astfel, toate celelalte transmisii (nesincronizate) sunt percepute ca un zgomot aleator. Dacă ele nu sunt prea numeroase, algoritmul fundamental de decodificare funcționează, totuși, destul de bine. Există multă teorie referitoare la suprapunerea secvenței de felii peste nivelul de zgomot (Pickholtz, și alții 1982). Așa cum vă puteți aștepta, cu cât secvența de felii este mai lungă, cu atât este mai mare probabilitatea detectării ei corecte în prezența zgomotului. Pentru o mai mare siguranță, secvența de biți poate folosi un cod corector de erori. Secvențele de felii nu folosesc niciodată coduri corectoare de erori.

O presupunere implicită în discuția anterioară este că nivelurile de putere ale tuturor stațiilor sunt aceleași cu cele percepute de receptor. Protocolul CDMA este folosit în mod obișnuit pentru sistemele fără fir cu o stație de bază fixă și multe stații mobile la distanțe variabile de aceasta. Nivelurile de putere, recepționate la stația de bază depind de cât de departe sunt emițătorii. În acest caz, o euristică bună este ca fiecare stație mobilă să emită către stația de bază la un nivel complementar de putere față de cel primit de la stația de bază. Altfel spus, o stație mobilă care recepționează un semnal slab va folosi mai multă putere decât una care primește un semnal puternic. De asemenea, stația de bază dă comenzi explicite către stațiile mobile pentru a-și crește/descrește puterea de transmisie.

De asemenea, am presupus că receptorul știe cine este emițătorul. În principiu, dată fiind suficientă putere de calcul, receptorul poate asculta toți emițătorii în același timp, prin rularea algoritmului de decodificare pentru fiecare dintre ei în paralel. În viața reală, este de ajuns să spunem că e mai ușor de zis decât de făcut. De asemenea, CDMA are mulți alți factori care complică soluția și care au fost comentați în această scurtă introducere. Totuși, CDMA este o schemă inteligentă care este rapid introdusă pentru comunicații mobile fără fir. În mod normal, operează într-o bandă de 1.25

MHz (față de 30 KHz pentru D-AMPS și 200 KHz pentru GSM), dar suportă în aceeași bandă mai mulți utilizatori decât oricare dintre celelalte sisteme. În practică, lărgimea de bandă disponibilă pentru fiecare utilizator este cel puțin la fel de bună ca la GSM și deseori mult mai bună.

Inginerii care doresc să înțeleagă foarte bine CDMA ar trebui să citească (Lee și Miller, 1998). O schemă alternativă de împrăștiere, în care împrăștierea se face în timp, și nu în frecvență, este descrisă în (Crespo et al., 1995). Încă o schemă este descrisă în (Sari et al., 2000). Toate aceste referințe necesită ceva cunoștințe în ingineria comunicațiilor.

2.6.3 A treia generație de telefoane mobile: Voce digitală și date

Care este viitorul în telefonia mobilă? Să aruncăm o privire rapidă. Un număr de factori dirijează industria. În primul rând, traficul de date depășește deja traficul de voce pe rețeaua fixă și crește exponențial, pe când traficul de voce este constant. Mulți experți din industrie se așteaptă ca traficul de date să domine traficul de voce și pe dispozitivele mobile, cât de curând. În al doilea rând, industriile telefonice, de divertisment și de calculatoare au devenit toate digitale și converg rapid. Mulți oameni își doresc un dispozitiv ușor, portabil care să se comporte ca un telefon, CD player, DVD player, terminal pentru poșta electronică, interfață Web, stație pentru jocuri, procesor de text, și chiar mai mult, totul însoțit de conectivitate internațională fără fir la Internet cu lărgime de bandă ridicată. Dispozitivul și modul de conectare al acestuia sunt ceea ce înseamnă generația a treia de telefonie mobilă. Pentru mai multe informații, vezi (Huber et al., 2000; și Sarikaya, 2000).

În 1992, ITU a încercat să fie mai precis în ceea ce privește acest vis și a inițiat un plan pentru a-și atinge scopul, numit **IMT-2000,** unde IMT înseamnă **International Mobile Telecommunications** (rom: Telecomunicații Mobile Internaționale). Numărul 2000 vine de la trei lucruri: (1) anul în care trebuia să intre în utilizare, (2) frecvența la care trebuia să funcționeze (în MHz), și (3) lărgimea de bandă pe care serviciul trebuia să o aibă (în KHz).

Nu a reușit în nici unul dintre aspectele propuse. Nimic nu a fost implementat până în 2000. ITU a recomandat ca toate guvernele să rezerve spectru la 2 GHz, astfel încât dispozitivele să comunice fără redirectări între țări. China a rezervat respectiva bandă de frecvență, dar nimeni altcineva nu mai făcut la fel. În sfârșit, a fost recunoscut că 2 Mbps nu este în prezent posibil pentru utilizatori care sunt prea mobili (datorită dificultății de a realiza schimburile suficient de repede). Mai realist este 2 Mbps pentru utilizatori staționari, în spații închise (ceea ce va rivaliza direct cu ADSL), 384 Kbps pentru oameni care merg, și 144 Kbps pentru conexiuni din mașini. Oricum, întreaga arie a **3G**, după cum este denumită, este un larg domeniu de activitate. A treia generație se poate să fie un pic mai puțin decât se aștepta și un pic mai târzie, dar este sigur că va apărea.

Principalele servicii pe care rețeaua IMT-2000 ar trebui să le ofere utilizatorilor săi sunt:

1. Transmisie de voce de înaltă calitate.
2. Mesagerie (înlocuirea poștei electronice, fax-ului, SMS-ului, chat-ului etc.).
3. Multimedia (muzică, vizualizare video, filme, televiziune) .
4. Acces la Internet (navigare pe Web, inclusiv pagini cu audio și video).

Servicii adiționale ar putea fi videoconferințele, prezentările televizate, jocurile în grup și comerțul mobil (fluturarea mobilului către casieră pentru a plăti într-un magazin). Mai mult, toate aceste servicii ar trebui să fie disponibile oriunde pe glob (prin conectare automată printr-un satelit atunci când nici o rețea terestră nu poate fi localizată), instant (întotdeauna conectat), și cu garanții pentru calitatea serviciului.

ITU a imaginat o singură tehnologie internațională pentru IMT-2000, astfel încât fabricanții să poată construi un singur dispozitiv care să fie vândut și folosit oriunde în lume (cum ar fi CD playerele și calculatoarele și nu cum sunt telefoanele mobile și televizoarele). O singură tehnologie ar face viața mult mai simplă și pentru operatorii de rețea și ar încuraja mai mulți oameni să folosească serviciile. Războaiele de format, cum a fost cel dintre Betamax și VHS atunci când au apărut videocasetofoanele, nu sunt benefice pentru afaceri.

Au fost făcute câteva propuneri, iar după mai multe trieri au rămas în discuție două propuneri mai interesante. Prima, **W-CDMA,** adică **Wideband CDMA** (rom: CDMA de bandă largă), a fost propusă de Ericsson. Acest sistem folosește secvență directă în spectru larg de tipul descris mai sus. Funcționează la o lărgime de bandă de 5 MHz și a fost proiectat să interacționeze cu rețele GSM actuale, fără a încerca să includă și compatibilitatea cu versiuni GSM anterioare. Are, totuși, proprietatea că un apelant poate părăsi o celulă W-CDMA și poate intra într-o celulă GSM fără a pierde apelul. Acest sistem a fost puternic susținut de Uniunea Europeană, care l-a numit **UMTS** (**Universal Mobile Telecommunications System**, rom: sistem universal de telecomunicații mobile) .

Cealaltă propunere a fost **CDMA2000**, propus de Qualcomm. Și acesta este un proiect bazat pe secvența directă în spectru larg, fiind de fapt o extensie a lui IS-95 și compatibil cu acesta. Folosește de asemenea o bandă de frecvență de 5 MHz, dar nu a fost proiectat să interacționeze cu GSM și nu poate redirecta apeluri către o celulă GSM (sau o celulă D-AMPS). Alte diferențe tehnice față de W-CDMA sunt vitezele de cip diferite, timpii de cadre diferiți, spectru diferite și moduri diferite de sincronizare.

Dacă inginerii de la Ericsson și de la Qualcomm ar fi fost închiși într-o singură cameră și li s-ar fi spus să găsească o soluție comună, probabil că ar fi reușit. De fapt, principiul de bază din spatele ambelor sisteme este CDMA într-un canal de 5 MHz și nimeni nu vrea să moară pentru viteza de cip preferată. Necazul este că problema reală nu este ingineria, ci politica (ca de obicei). Europa a vrut un sistem care să poată interacționa cu GSM, iar S.U.A. doreau un sistem care să fie compatibil cu unul deja popular în Statele Unite (IS-95). În plus, fiecare parte a sprijinit compania locală (Ericsson este originară din Suedia, iar Qualcomm este din California). În fine, Ericsson și Qualcomm au fost implicate în numeroase procese privind patentele asupra respectivelor proiecte CDMA.

În martie 1999, cele două companii au terminat cu procesele atunci când Ericsson a fost de acord să cumpere infrastructura Qualcomm. S-au înțeles de asemenea asupra unui singur standard 3G, dar unul cu multe opțiuni incompatibile, care până la urmă recunoaște pe hârtie diferențele tehnice. Aceste dispute fiind încheiate, dispozitivele 3G și serviciile ar putea începe să apară în viitorii ani.

Multe s-au scris despre sistemele 3G, multe lucrări lăudându-le ca pe cel mai mare lucru de la pâinea feliată încoace. Unele referințe sunt (Collins și Smith, 2001; De Vriendt et al., 2002; Harte et al., 2002; Lu, 2002 și Sarikaya, 2000). Totuși, unii pesimiști încă mai cred că industria se îndreaptă în direcția greșită (Garber, 2002 și Goodman, 2000).

În așteptarea sfârșitului disputei asupra 3G, unii operatori fac cu prudență un pas mic și atent în direcția 3G prin a merge spre ceea ce este numit uneori **2.5G**, deși 2.1G este mai exact. Un astfel de sistem este **EDGE** (**Enhanced Data rates for GSM Evolution**, rom: viteze de transfer de date mărite pentru evoluția GSM), ceea ce este chiar GSM cu mai mulți biți pe baud. Problema este că mai mulți biți pe baud înseamnă mai multe erori pe baud, astfel că EDGE dispune de nouă scheme diferite de modulare și corectare de erori, în funcție de ce parte din bandă este dedicat corectării erorilor introduse de viteza mai mare.

O altă schemă 2.5G este **GPRS** (**General Packet Radio Service**, rom: serviciul radio pentru pachete generice), care este o rețea de pachete generale peste D-AMPS sau GSM. Ea permite stațiilor

mobile să trimită și să primească pachete IP într-o celulă folosind sistemul pentru voce. În timpul operării GPRS, unele intervale de timp ale unor frecvențe sunt rezervate pentru traficul de pachete. Numărul și poziția intervalelor pot fi administrate dinamic de către stația de bază, în funcție de raportul între de trafic de voce și de date din celulă.

Intervalele de timp disponibile sunt împărțite în câteva canale logice, folosite în diverse scopuri. Stația de bază determină alocarea de canale logice peste intervalele de timp. Un canal logic este utilizat pentru preluarea de pachete de la stația de bază la o stație mobilă, fiecare pachet indicând cui îi este destinat. Pentru a trimite un pachet IP, o stație mobilă solicită unul sau mai multe intervale de timp prin trimiterea unei cereri la stația de bază. Dacă această cerere ajunge fără probleme, stația de bază anunță frecvența și intervalele de timp alocate mobilului pentru a trimite pachetul. O dată ce pachetul a ajuns la stația de bază, este transferat în Internet printr-o conexiune cu fir. Deoarece GPRS este doar un nivel suplimentar suprapus peste sistemul de voce deja existent, el reprezintă în cel mai bun caz o soluție de umplere a golului până la sosirea lui 3G.

Deși rețelele 3G nu sunt complet dezvoltate deocamdată, unii cercetători privesc 3G ca pe o afacere încheiată și, deci, care nu mai prezintă interes. Acești oameni lucrează deja la sistemele 4G (Berezdivin et al., 2002; Guo și Chaskar, 2002; Huang și Zhuang, 2002; Keller et al., 2002; și Misra et al., 2002). Unele dintre caracteristicile propuse pentru sistemele 4G includ lărgime de bandă mare, omniprezență (conectivitate oriunde), integrare simplă cu rețelele cablate și mai ales cu IP, administrare de resurse și spectru, transmisii radio software și calitatea serviciilor pentru aplicații multimedia.

Pe de altă parte, sunt amenajate la tot pasul o sumedenie de puncte de acces pentru LAN-urile fără fir 802.11, astfel încât unii oameni cred că 3G este nu numai o afacere neîncheiată, ci și că este condamnat la dispariție. În această viziune, oamenii se vor plimba de la un punct de acces 802.11 la altul pentru a rămâne conectați. A spune că „industria este într-o continuă creștere" este doar o imensă subestimare. Reveniți la această discuție peste 5 ani, ca să vedeți ce s-a întâmplat.

2.7 TELEVIZIUNEA PRIN CABLU

Am studiat până acum atât sistemele de telefonie fixă cât și pe cele de telefonie fără fir în mod egal. Ambele vor juca un rol important în rețelele viitorului. Totuși, o nouă alternativă disponibilă pentru rețelele fixe devine acum un jucător important: rețelele de televiziune prin cablu. Mulți oameni primesc deja telefonul și serviciile Internet prin cablu, iar operatorii de cablu muncesc din greu pentru a-și crește cota pe piață. În următoarele paragrafe vom privi mai în detaliu televiziunea prin cablu ca sistem de rețele și îl vom compara cu sistemele de telefonie pe care tocmai le-am studiat. Pentru mai multe informații despre televiziunea prin cablu, vezi (Laubach et al.,2001; Louis, 2002; Ovadia,2001; și Smith, 2002).

2.7.1 Televiziune prin antena colectivă

Televiziunea prin cablu fost concepută la sfârșitul anilor 1940 ca un mijloc de a oferi recepție mai bună oamenilor care trăiau în zone rurale sau muntoase. Inițial, sistemul era format dintr-o antenă mare montată pe vârful unui deal pentru a capta semnalul de televiziune din aer, un amplificator,

numit **amplificator terminal** (eng: **head end**), pentru a spori puterea semnalului, și un cablu coaxial pentru a transmite semnalul în casele oamenilor, după cum este ilustrat în fig. 2-46.

În primii ani, televiziunea prin cablu era numită t**eleviziune prin antena colectivă**. Se asemăna foarte mult cu o operație mama-și-tata; oricine era destul de îndemânatic la electronică putea pune la punct un serviciu pentru orașul lui, pentru ca apoi utilizatorii să participe la plata costurilor. Deoarece numărul de abonați a crescut, au trebuit adăugate cabluri în derivație din cablul original și s-au adăugat noi amplificatoare acolo unde a fost nevoie de ele. Transmisia era într-un singur sens, de la amplificatorul din capăt, de lângă antenă, spre utilizatori. Până în 1970, existau mii de astfel de sisteme independente.

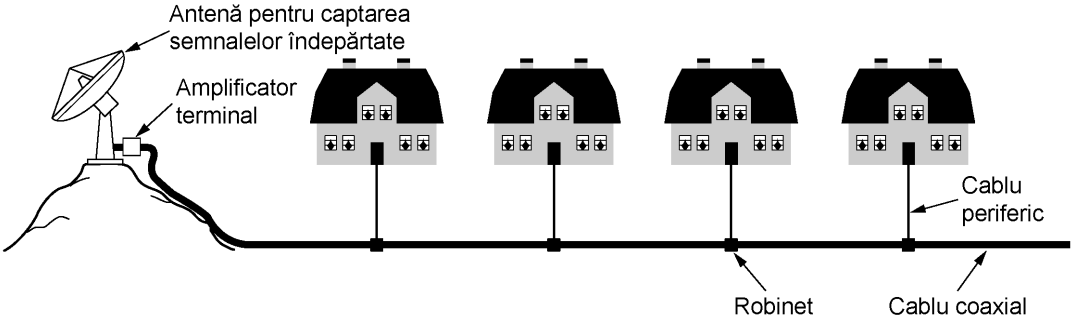


Fig. 2-46. Un sistem primitiv de televiziune prin cablu

În 1974, Time Inc., a pus bazele unui nou canal, Home Box Office, cu conținut nou (filme) și distribuit doar prin cablu. Au urmat alte canale distribuite doar pe cablu, cu știri, sport, gastronomie și multe alte subiecte. Această dezvoltare a dat naștere la două schimbări în industrie. În primul rând, mari corporații au început să cumpere sisteme de cablu deja existente și să întindă noi cabluri pentru a atrage noi abonați. În al doilea rând, exista acum cerința de a se conecta mai multe sisteme de cablu, adesea aflate în orașe îndepărtate, pentru a fi distribuite canalele noi de televiziune prin cablu. Companiile de cablu au început să întindă cabluri între orașele lor pentru a le conecta pe toate într-un sistem unic. Acest model a fost analog cu ceea ce s-a întâmplat în industria telefonică în urmă cu 80 de ani în cazul conectării oficiilor periferice izolate pentru a face posibile apelurile la distanță mare.

2.7.2 Internet prin cablu

De-a lungul anilor sistemul de cablu a crescut și cablurile dintre diversele orașe au fost înlocuite cu fibră cu lărgime de bandă mare, similar cu ceea ce s-a întâmplat în sistemul telefonic. Un sistem cu fibră pentru transportul pe distanțe lungi și cu cablu coaxial până la casele clienților este numit sistem **HFC** (**Hybrid Fiber Coax**, rom: hibrid fibră–coaxial). Convertoarele electro-optice folosite pentru interfațarea părților optice și electrice ale sistemului sunt numite **noduri de fibră**. Pentru că lărgimea de bandă a fibrei este mult mai mare decât cea a cablului coaxial, un nod de fibră poate alimenta mai multe cabluri coaxiale. O parte a unui sistem HFC modern este prezentat în fig. 2-47(a).

În ultimii anii, mulți operatori de cablu au decis să intre în afaceri legate de accesul la Internet, și adesea și în afaceri legate de telefonie. Totuși, diferențele tehnice dintre centralele de cablu și cele de telefon au efecte semnificative asupra operațiilor care trebuie făcute pentru a atinge aceste scopuri. De exemplu, toate amplificatoarele unidirecționale din sistem trebuie înlocuite cu amplificatoare bidirecționale.

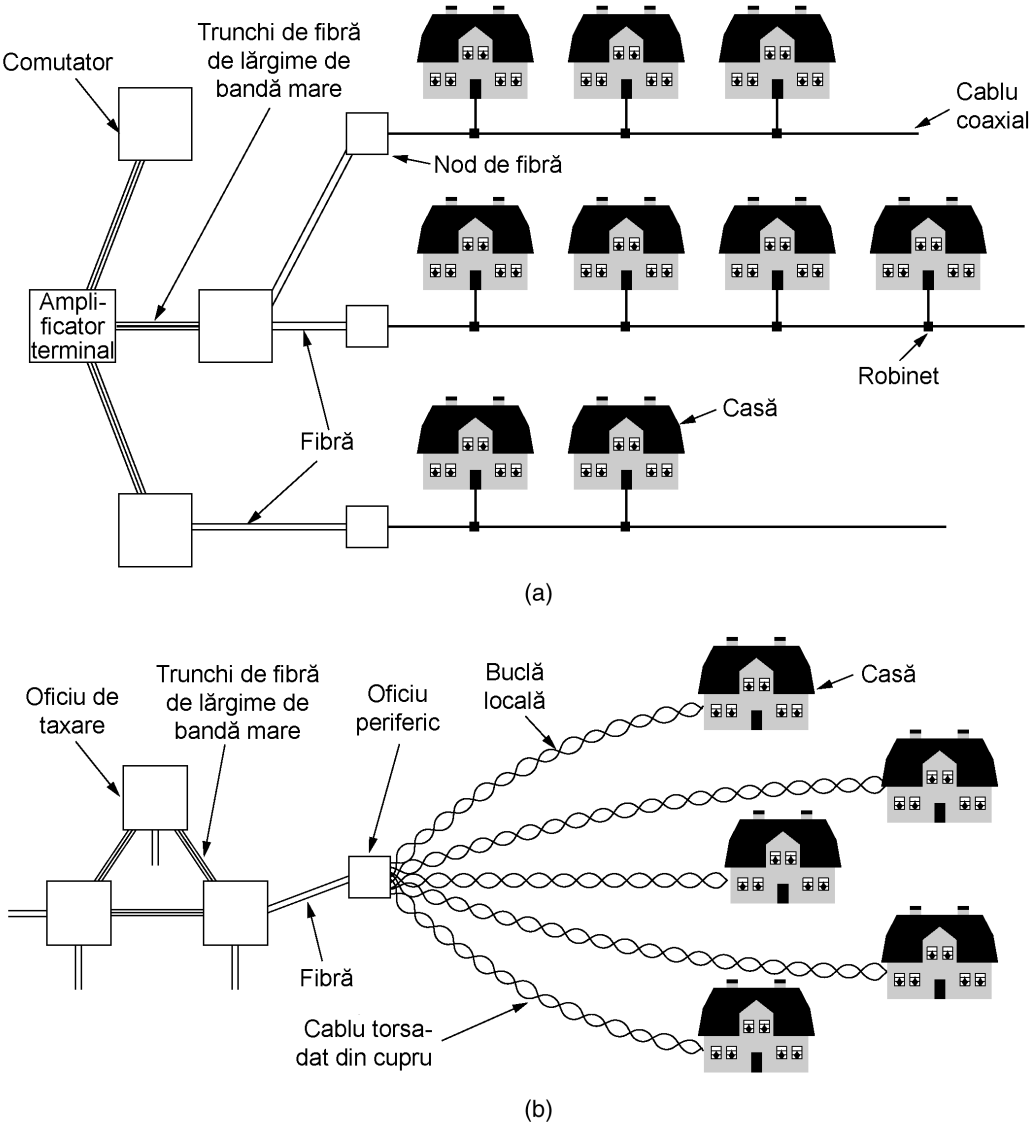


Fig. 2-47. (a) Televiziune prin cablu. (b) Sistemul de telefonie fixă.

Totuși, există între sistemul HFC din fig. 2-47(a) și sistemul telefonic din fig. 2-47(b) o diferență care este mult mai greu de înlăturat. În inima cartierelor, un singur cablu este utilizat în comun de mai multe case, pe când în sistemul telefonic, fiecare casă are propria buclă locală. Când este folosit pentru difuzarea televiziunii, această utilizare multiplă nu joacă nici un rol. Toate programele sunt difuzate pe cablu și nu are nici o importanță dacă sunt 10 telespectatori sau 10.000 de telespectatori. Când același cablu este utilizat pentru acces la Internet, contează foarte mult dacă sunt 10 utilizatori sau 10.000. Dacă unul dintre utilizatori se decide să preia de pe Internet un fișier foarte mare, lărgimea de bandă respectivă este teoretic luată de la alți utilizatori. Cu cât sunt mai mulți utilizatori, cu atât este mai mare competiția pentru lărgimea de bandă. Sistemul telefonic nu are această caracte-

ristică: salvarea unui fișier foarte mare prin intermediul unei linii ADSL nu reduce banda de frecvență a vecinului. Pe de altă parte, lărgimea de bandă a cablului coaxial este mult mai mare decât cea a perechilor torsadate.

Soluția prin care industria cablului a rezolvat această problemă a fost separarea cablurilor lungi și conectarea fiecăreia dintre bucăți direct la un nod de fibră. Lărgimea de bandă de la amplificator până la fiecare nod de fibră este teoretic infinită, așa că atâta timp cât nu sunt foarte mulți abonați pe fiecare segment de cablu, cantitatea de trafic este rezonabilă. În prezent, cablurile obișnuite deservesc 500-2000 de case. Din ce în ce mai mulți oameni se abonează la Internet prin cablu, așa că încărcarea ar putea să devină prea mare, necesitând mai multe separări și mai multe noduri de fibră.

2.7.3 Alocarea de spectru

Renunțarea la toate canalele TV și utilizarea infrastructurii de cablu strict pentru accesul la Internet ar genera probabil un număr considerabil de clienți iritați, astfel încât companiile de cablu ezită să facă acest lucru. Mai mult, cele mai multe orașe cenzurează serios ceea ce este pe cablu, astfel că operatorii de cablu nu ar putea face acest lucru nici dacă ar dori într-adevăr. Ca o consecință, a fost nevoie să se găsească o modalitate ca televiziunea si Internetul sa coexiste pe același cablu.

Canalele televiziunii prin cablu din America de Nord ocupă în mod normal regiunea de 54-550 MHz (mai puțin regiunea de la 88 la 108 MHz, atribuită radioului FM). Aceste canale au 6 MHz lățime, inclusiv benzile de siguranță. În Europa, limita inferioară este de obicei de 65 MHz și canalele au 6-8 MHz lățime pentru rezoluția mai înaltă cerută de PAL și SECAM, dar altfel schema de alocare este similară. Partea de jos a benzii nu este folosită. De asemenea, cablurile moderne pot opera cu mult peste 550 MHz, adesea la 750 MHz sau mai mult. Soluția aleasă a fost introducerea de canale ascendente în banda 5-42 MHz (ceva mai sus în Europa) și utilizarea frecvențelor de la limita superioară pentru canale descendente. Spectrul cablului este ilustrat în fig. 2-48.

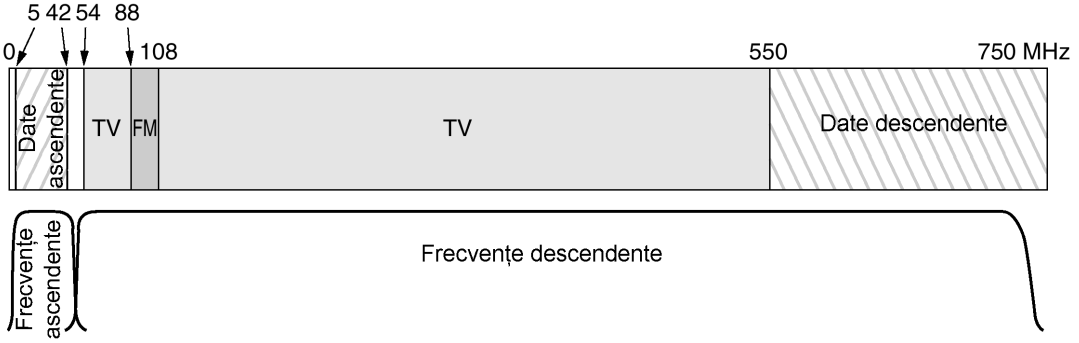


Fig. 2-48. Alocarea frecvențelor într-un sistem tipic de TV prin cablu folosit pentru acces la Internet

Observați că din momentul în care semnalele de televiziune sunt toate descendente, este posibil să se folosească amplificatoare ascendente care funcționează doar în regiunea 5-42 MHz și amplificatoare descendente care funcționează doar la 54 MHz și mai sus, după cum se vede în figură. Astfel, obținem o asimetrie între benzile ascendente și descendente, pentru că este disponibil mai mult spectru deasupra spectrului televiziunii, decât sub acesta. Pe de altă parte, mare parte din trafic este probabil descendent, astfel că operatorii de cablu nu sunt supărați din această cauză. După cum am

văzut mai devreme, și companiile telefonice oferă de obicei un serviciu DSL asimetric, chiar dacă nu au nici un motiv tehnic să facă acest lucru.

Cablurile coaxiale lungi nu sunt cu nimic mai bune în transmiterea semnalelor digitale decât sunt buclele locale lungi, astfel că modularea analogică este necesară și aici. Schema uzuală este să se ia fiecare canal descendent de 6 MHz sau 8 MHz și să se moduleze cu QAM-64 sau, pentru o calitate excepțională a transmisiei prin cablu, QAM-256. Cu un canal de 6 MHz și QAM-64 obținem 36 Mbps. După ce scădem supraîncărcarea, informația utilă netă este de aproape 27 Mbps. Cu QAM-256, informația utilă netă este aproape 39 Mbps. Valorile europene sunt cu 1/3 mai mari.

Pentru canalele ascendente, chiar și QAM-64 nu funcționează prea bine. Este prea mult zgomot de la microundele terestre, radiourile CB și alte surse, astfel că se folosește o schemă mai conservativă – QPSK. Această metodă (prezentată în fig. 2-25) generează 2 biți pe baud în loc de 6 sau 8 biți pe care QAM îi furnizează pe canalele descendente. În consecință, asimetria dintre banda descendentă și cea ascendentă este mai profundă decât sugerează fig. 2-48.

Pe lângă actualizarea amplificatoarelor, operatorul trebuie să actualizeze și capătul de pornire al cablului, transformându-l dintr-un amplificator primitiv într-un sistem computerizat digital inteligent cu o interfață de lărgime de bandă mare între fibră și ISP. Adesea și numele este actualizat, din amplificator terminal al cablului în **CTMS** (**Cable Modem Termination System**, rom: sistem terminal pentru modemuri de cablu). În continuarea acestui text, vom evita să facem o actualizare de nume și vom menține tradiționalul amplificator terminal al cablului (headend).

2.7.4 Modemuri de cablu

Accesul la Internet prin cablu necesită un modem pentru cablu, dispozitiv care conține două interfețe: una către calculator și una către rețeaua de cablu. În primii ani ai Internetului prin cablu, fiecare operator avea un modem propriu patentat pentru cablu, care era instalat de un tehnician al companiei de cablu. Totuși, a devenit în curând evident că un standard ar crea o piață competitivă de modemuri pentru cablu și ar scădea prețurile, încurajând astfel utilizarea serviciului. Mai mult, clienții care cumpără modemurile pentru cablu din magazin și le instalează singuri (la fel cum procedează cu modemurile telefonice V.9x) ar putea scuti firma de cheltuielile intervențiilor la domiciliu.

În consecință, operatorii de cablu mai importanți au făcut echipă cu o companie numită CableLabs pentru a produce un standard de modem de cablu și pentru a testa produsele din punct de vedere al compatibilității. Acest standard, numit **DOCSIS** (**Data Over Cable Service Interface Specification**, rom: specificațiile interfeței serviciului de date prin cablu) abia începe să înlocuiască modemurile patentate. Versiunea europeană se numește **EuroDOCSIS**. Nu tuturor operatorilor de cablu le place ideea unui standard, deoarece mulți dintre ei au făcut bani frumoși prin închirierea modemurilor lor către clienții lor captivi. Un standard public cu zeci de fabricanți care vând modemuri de cablu în magazine ar pune punct acestei practici profitabile.

Interfața modem-calculator este directă. Uzual, este 10 Mbps Ethernet (sau ocazional USB) în prezent. În viitor, întregul modem ar putea fi un card mic, conectat direct la calculator, la fel cum sunt modemurile interne V.9x .

Celălalt capăt este mai complicat. O mare parte a standardului se referă la ingineria radio, un subiect care este cu mult în afara scopului acestei cărți. Singura parte care merită să fie menționată aici este aceea că modemurile pentru cablu, ca și modemurile ADSL, sunt întotdeauna conectate. Ele deschid o conexiune când sunt activate și mențin acea conexiune atât timp cât sunt alimentate pentru că operatorii de cablu nu percep taxe diferite în funcție de durata conexiunii.

Pentru a înțelege mai bine cum funcționează aceste modemuri, să vedem ce se întâmplă atunci când un modem de cablu este conectat la rețeaua electrică și pornit. Modemul scanează canalele descendente căutând un pachet special, este trimis periodic de către alimentatorul terminal (headend) pentru a furniza parametrii sistemului către modemurile care tocmai s-au conectat. După ce detectează acest pachet, noul modem își anunță prezența pe unul dintre canalele ascendente. Amplificatorul terminal răspunde alocându-i modemului canalele corespunzătoare pentru flux ascendent și descendent. Aceste alocări pot fi schimbate mai târziu dacă amplificatorul terminal consideră că este necesar să echilibreze încărcarea.

Modemul determină apoi distanța de la amplificatorul terminal trimițând către acesta un pachet special și măsurând cât îi ia să primească răspunsul. Acest proces se numește **ranging** (rom: poziționare). Este important pentru modem să cunoască aceasta distanță pentru a regla modul de funcționare a canalelor de flux ascendent și pentru a reuși sincronizarea. Aceste canale sunt divizate în timp în **mini-intervale** (eng. minislots). Fiecare pachet de flux ascendent trebuie să încapă într-unul sau mai multe mini-intervale consecutive. Amplificatorul terminal anunță periodic începutul unei noi serii de mini-intervale, dar pistolul de start nu se aude simultan la toate modemurile, din cauza timpului de propagare prin cablu. Cunoscând cât de departe se află de capăt, fiecare modem poate să calculeze cu cât timp în urmă a început cu adevărat primul mini-interval. Lungimea mini-intervalelor este dependentă de rețea. O încărcare tipică este de 8 octeți.

În timpul inițializării, amplificatorul terminal alocă fiecărui modem și câte un mini-interval pentru a cere lățime de bandă de flux ascendent. Ca o regulă, mai multor modemuri li se va aloca același mini-interval, ceea ce duce la conflicte. Atunci când un calculator vrea să transmită un pachet, el transferă pachetul modemului, care cere apoi numărul de mini-intervale necesare pentru el. Dacă această cerere este acceptată, amplificatorul terminal transmite o confirmare pe canalul de flux descendent, spunându-i modemului ce mini-intervale au fost rezervate pentru pachetul său. Pachetul este apoi trimis, începând de la mini-intervalul alocat lui. Pentru a solicita transmisia de pachete adiționale se folosește un câmp din antet.

Pe de altă parte, dacă apare un conflict pentru mini-intervalul cerut, nu va exista nici o confirmare, iar modemul va aștepta un timp oarecare și apoi va încerca din nou. După fiecare eșec succesiv, timpul aleator maxim se dublează. (Pentru cititorii deja familiarizați într-o oarecare măsură cu rețelele, acest algoritm este ALOHA discretizat cu regresie exponențială binară. Nu se poate folosi Ethernet pe cablu pentru că stațiile nu pot asculta mediul. Vom reveni la aceste teme în Cap.4).

Canalele descendente sunt administrate diferit față de cele ascendente. În primul rând, avem un singur emițător (amplificatorul terminal), deci nu apar conflicte și nu este nevoie de mini-intervale, care reprezintă de fapt o multiplexare statistică prin divizarea timpului. În al doilea rând, traficul descendent este de obicei mult mai mare decât cel ascendent, astfel încât se folosește o dimensiune de pachet fixată la 204 octeți. O parte din acest pachet reprezintă codul corector de erori Reed-Solomon și alte informații suplimentare, lăsând utilizatorului un pachet util de 184 octeți. Aceste numere au fost alese pentru compatibilitate cu televiziunea digitală care folosește MPEG-2, astfel încât canalele TV și canelele de flux descendent de date să fie formatate identic. La nivel logic, conexiunile sunt prezentate în fig. 2-49.

Revenind la inițializarea modemului, o data ce un modem a terminat poziționarea și și-a căpătat canalul de flux ascendent, canalul de flux descendent și alocarea mini-intervalelor, este liber să înceapă să trimită pachete. Primul pachet este trimis către ISP cerând o adresă IP, care este alocată dinamic folosind un protocol numit DHCP, pe care îl vom studia în cap. 5. De asemenea cere și primește ora exactă de la amplificatorul terminal.

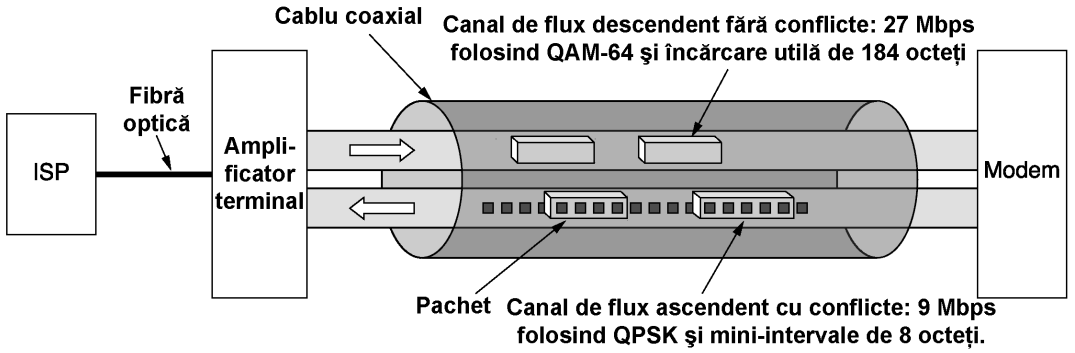

Cablu coaxial
Canal de flux descendent fără conflicte: 27 Mbps
folosind QAM-64 şi încărcare utilă de 184 octeți
Fibră optică
ISP
Ampli-ficator terminal
Modem
Pachet
Canal de flux ascendent cu conflicte: 9 Mbps
folosind QPSK şi mini-intervale de 8 octeți.

Fig. 2-49. Detaliile tipice pentru canalele de flux de date ascendent și descendent în America de Nord

Următorul pas implică securitatea. Cablul fiind un mediu partajat, oricine își dă silința poate să citească tot traficul care trece pe lângă el. Pentru a preveni situația în care oricine poate să își asculte vecinii (la propriu), tot traficul este criptat în ambele direcții. O parte a procedurii de inițializare presupune stabilirea cheilor de criptare. La început cineva ar putea să creadă că a avea doi străini, amplificatorul terminal și modemul care să stabilească o cheie secreta în plină zi cu mii de oameni uitându-se la ei ar fi imposibil. Se va demonstra că nu e așa, dar trebuie să așteptăm până în Cap. 8 ca să explicam cum (răspunsul, pe scurt: folosind algoritmul Diffie-Hellman).

În sfârșit, modemul trebuie să se conecteze și să ofere identificatorul său unic prin canalul sigur. În acest moment, inițializarea este completă. Utilizatorul poate să se conecteze la ISP și să treacă la treabă.

Se mai pot spune multe lucruri despre modemurile de cablu. Unele referințe relevante sunt (Adams și Dulchinos, 2001; Donaldson și Jones, 2001 și Dutta-Roy, 2001).

2.7.5 Comparație între ADSL și cablu

Care variantă este mai bună, ADSL sau cablul? Este ca și cum ai întreba care sistem de operare este mai bun. Sau care limbă este mai bună. Sau care religie. Răspunsul primit depinde de persoana întrebată. Să comparăm ADSL și cablul din câteva puncte de vedere. Ambele folosesc fibra optică drept coloană vertebrală, dar diferă la periferie. Sistemele prin cablu folosesc coaxial; ADSL folosește perechi torsadate. Capacitatea teoretică de transport a cablului este de sute de ori mai mare decât cea a perechii torsadate. Totuși, nu este disponibilă întreaga capacitate a cablului pentru utilizatorii de date, pentru că mare parte din lățimea de bandă este irosită de chestiuni nefolositoare cum sunt programele de televiziune.

Practic, este greu să facem o estimare generală pentru capacitatea efectivă. Furnizorii de ADSL fac anumite declarații despre lățimea de bandă (de exemplu, 1 Mbps flux descendent, 256 Kbps flux ascendent) și în general reușesc să onoreze în mod consistent cam 80 % din acestea. Furnizorii de cablu nu pot să declare nimic, deoarece capacitatea efectivă depinde de câți utilizatori sunt activi în acel moment pe segmentul de cablu al utilizatorului. Uneori poate fi mai bine decât cu ADSL, alteori poate fi mai rău. Ceea ce ar putea fi deranjant, totuși, este tocmai această variație imprevizibilă a calității serviciilor. Faptul că ai servicii excelente acum nu garantează servicii excelente peste un minut, deoarece se poate ca marele devorator de lărgime de bandă din oraș să-și fi pornit calculatorul.

Chiar dacă un sistem ADSL câștigă mai mulți utilizatori, creșterea numărului lor nu are un efect semnificativ asupra utilizatorilor existenți, pentru că fiecare utilizator are o conexiune dedicată. În cazul cablului, pe măsură ce mai mulți clienți se abonează la servicii Internet, performanța pentru utilizatorii existenți va scădea. Singurul remediu pentru operatorul de cablu este să separe cablurile aglomerate și să conecteze fiecare dintre ele direct la un nod de fibra optică. Acest lucru costă timp și bani, deci există presiuni financiare pentru a-l evita.

Ca o paranteză, am studiat deja un alt sistem bazat, la fel ca și cablul, pe folosirea unui canal partajat: sistemul telefonic mobil. Și aici, un grup de utilizatori, pe care i-am putea numi colegi de celula, împart o cantitate fixă de lărgime de bandă. În mod normal, aceasta este divizată prin FDM și TDM în felii fixe, alocate pentru utilizatorii activi, traficul de voce fiind destul de uniform. Dar pentru traficul de date, aceasta împărțire rigidă este foarte ineficientă, deoarece utilizatorii de date sunt deseori în așteptare, caz în care banda rezervată lor este irosită. Totuși, din acest punct de vedere, accesul prin cablu se aseamănă mai mult cu sistemul de telefonie mobilă decât cu acela de telefonie fixă.

Disponibilitatea serviciului este un aspect în care ADSL și cablul diferă. Toată lumea are un telefon, dar nu toți utilizatorii sunt destul de aproape de oficiul final pentru a obține ADSL. Pe de altă parte, nu toata lumea are televiziune prin cablu, dar dacă ai cablu și compania ta de cablu oferă acces Internet, poți să te abonezi. Distanța până la nodul de fibră optică sau până la amplificatorul terminal nu reprezintă o problemă. De asemenea, merită menționat că, deoarece cablul a pornit ca un mediu de distribuție al televiziunii, puține companii folosesc așa ceva.

Fiind un mediu punct-la-punct, ADSL este prin definiție mai sigur decât cablul. Orice utilizator de cablu poate citi cu ușurință pachetele ce vin pe cablu. Din acest motiv, orice furnizor de cablu decent va cripta traficul în ambele direcții. Oricum, chiar dacă știi că acela care îți citește mesajele criptate este vecinul tău, tot este mai puțin sigur decât dacă nu ți le citea nimeni.

Sistemul telefonic este în general mai de încredere decât cablul. Dispune, de exemplu, de un sistem suplimentar de alimentare cu energie în caz de urgență și funcționează normal chiar și în timpul unei pene de curent. La cablu, dacă energia pentru orice amplificator de-a lungul lanțului cade, toți utilizatorii de după el sunt deconectați instantaneu.

În sfârșit, cei mai mulți furnizori ADSL oferă posibilitatea alegerii unui ISP. Uneori li se cere acest lucru chiar prin lege. Nu este întotdeauna cazul cu operatorii prin cablu.

Concluzia este că ADSL și cablul sunt mai mult asemănătoare decât diferite. Oferă servicii comparabile și, cum competiția dintre ei se încinge, probabil că vor oferi și prețuri comparabile.

2.8 REZUMAT

Nivelul fizic stă la baza tuturor rețelelor. Natura impune două limite fundamentale asupra unui canal, iar acestea determină lărgimea de bandă. Este vorba despre limita Nyquist, care se aplică asupra canalelor fără zgomot, și limita Shannon, pentru canale cu zgomot.

Mediile de transmisie pot fi ghidate sau neghidate. Principalele medii ghidate sunt cablul torsadat, cablul coaxial și fibra optică. Mediile neghidate includ undele radio, microundele, undele în infraroșu și razele laser care se propagă prin aer. Un sistem de transmisie readus în actualitate este comunicația prin satelit, în special sistemele LEO.

Elementul cheie pentru majoritatea rețelelor pe arii geografic largi este sistemul telefonic. Componentele sale principale sunt buclele locale, trunchiurile și comutatoarele. Buclele locale sunt circuite analogice de cablu torsadat, care necesită folosirea modemurilor pentru transmisia datelor digitale. ADSL oferă viteze de până la 50 Mbps prin divizarea buclei locale în mai multe canale virtuale și modularea fiecăruia separat. Buclele locale fără fir reprezintă o noua direcție de dezvoltare de urmărit, în special LMDS.

Trunchiurile sunt digitale și pot fi multiplexate în mai multe moduri, printre care: FDM, TDM și WDM. Atât comutarea de circuite cât și comutarea de pachete reprezintă tehnologii importante.

Pentru aplicațiile mobile, sistemul de telefonie fixă (cu fir) nu este adecvat. Radioul celular este folosit tot mai mult pentru transmisia de voce, iar în curând va fi folosit frecvent și pentru schimbul de date. Prima generație de mobile a fost analogică, dominată de AMPS. A doua generație a fost digitală, cu D-AMPS, GSM și CDMA ca opțiuni majore. A treia generație va fi digitală și bazată pe CDMA de bandă largă.

Un sistem alternativ pentru accesul la rețea este sistemul de televiziune prin cablu, care a evoluat gradat de la o antenă colectivă la un sistem hibrid fibra optică–coax. Potențial, acesta oferă lărgime de bandă foarte mare, dar lărgimea de bandă reală disponibilă depinde semnificativ de numărul de utilizatori activi la un moment dat și de ceea ce fac ei.

2.9 PROBLEME

1. Calculați coeficienții Fourier pentru funcția $f(t) = t$, $(0 \le t \le 1)$.

2. Un canal de 4 KHz, fără zgomot, este eșantionat la fiecare 1 msec. Care este rata maximă de transfer a datelor?

3. Canalele de televiziune au o lățime de 6 MHz. Câți biți/sec pot fi transmiși dacă se folosesc semnale digitale pe patru niveluri? Considerați cazul unui canal fără zgomot.

4. Dacă un semnal binar este transmis pe un canal de 3 KHz al cărui raport semnal-zgomot este de 20dB, care este rata maximă de transfer a datelor ce se poate realiza?

5. Ce raport semnal-zgomot este necesar pentru a pune o purtătoare T1 pe o linie de 50 KHz?

6. Care este diferența dintre o stea pasivă și un repetor activ într-o rețea pe fibră optică?

7. Care este lărgimea de bandă existentă în 0,1 microni de spectru la o lungime de undă de 1 micron?

8. Se dorește să se transmită prin fibră optică o secvență de imagini de pe ecranul calculatorului. Ecranul are 480 x 640 pixeli, fiecare pixel având 24 biți. Există 60 imagini ecran pe secundă. Ce lățime de bandă este necesară și care este lungimea de undă necesară pentru această bandă la 1,30 microni?

9. Teorema lui Nyquist este adevărată și pentru fibra optică sau numai pentru cablul de cupru?

10. În fig. 2-6 banda din partea stângă este mai îngustă decât celelalte. De ce?

11. Deseori, antenele radio funcționează cel mai bine atunci când diametrul antenei este egal cu lungimea de undă a undei radio. Antenele rezonabile au între 1 cm și 5 m în diametru. Ce domeniu de frecvență acoperă acestea?

12. Atenuarea multi-căi este maximizată atunci când două raze sosesc cu un defazaj de 180 grade. Cât de mare trebuie să fie diferența de drum pentru a maximiza atenuarea în cazul unei legături prin microunde de 1 GHz având 50 km lungime?

13. O rază laser de 1 mm lățime este urmărită de un detector de 1 mm lățime aflat la 100 m distanță, pe acoperișul unei clădiri. Care este limita maximă a deviației unghiulare (în grade) a laserului pentru care raza poate fi captată de detector ?

14. Cei 66 de sateliți de joasă altitudine din proiectul Iridium sunt împărțiți în 6 coliere în jurul Pământului. La altitudinea la care sunt folosiți, perioada de rotație este de 90 minute. Care este intervalul mediu pentru timpii morți (de inactivitate) în cazul unui emițător staționar?

15. Se considera un satelit la altitudinea sateliților geostaționari, dar al cărui plan orbital este înclinat față de planul ecuatorului cu un unghi ϕ. Un utilizator staționar se află pe suprafața pământului la latitudinea nordică ϕ. Este adevărat că acestui utilizator i se pare că satelitul este nemișcat pe cer? Dacă nu, descrieți mișcarea pe care o percepe.

16. Câte coduri de oficiu final erau înainte de 1984, atunci când fiecare oficiu final era identificat după codul său de zonă, format din 3 cifre, combinat cu primele trei cifre ale numărului local ? Codurile de zonă începeau cu o cifra din intervalul 2-9, avea a doua cifră un 0 sau un 1 și se puteau termina cu orice cifră. Primele două cifre din numărul local erau întotdeauna din intervalul 2-9. A treia cifră putea fi oricare.

17. Folosind *numai* datele din text, care este numărul maxim de telefoane pe care sistemul existent în SUA le poate suporta fără a se schimba numerotația și fără a se adăuga echipament adițional? Ar putea fi atins acest număr în realitate? Pentru simplificarea ipotezei, un calculator sau un fax este numărat tot ca un telefon. Presupuneți că exista un singur dispozitiv pentru o linie de abonat.

18. Un sistem telefonic simplu este alcătuit din două oficii finale și un singur oficiu de taxare, la care fiecare oficiu final este conectat printr-un trunchi duplex de 1 MHz. Un telefon obișnuit este folosit pentru a face 4 apeluri într-o zi lucrătoare de 8 ore. Durata medie a unui apel este de 6 minute. 10% dintre apeluri sunt de distanță lungă (adică traversează oficiul de taxare). Care este numărul maxim de telefoane pe care îl poate suporta un oficiu final? (presupuneți 4 KHz pe circuit)

19. O companie regională de telefoane are 10 milioane de abonați. Fiecare dintre telefoanele acestora este conectat la un oficiu central printr-un cablu torsadat de cupru. Lungimea medie a acestor cabluri este de 10 km. Cât de mult reprezintă cuprul din valoarea buclelor locale? Presupuneți că secțiunea transversală a fiecărui fir este de 1 mm diametru, greutatea specifică a cuprului este 9,0 și cuprul se vinde cu 3 dolari pe kg.

20. O conductă de petrol este un sistem simplex, half duplex, full duplex sau nici una dintre variantele menționate?

21. Costul unui microprocesor rapid a scăzut într-atât încât este posibil să se includă câte unul în fiecare modem. Cum afectează aceasta gestiunea erorilor liniei telefonice?

22. O diagramă-constelație a unui modem, similară celei din fig. 2-25, are puncte la următoarele coordonate: (1,1), (1,-1), (-1,1) și (-1,-1). Câți biți pe secundă poate atinge un modem cu acești parametri, la 1200 baud ?

23. O diagramă-constelație a unui modem, similară celei din fig. 2-25, are puncte în (0,1) și (0,2). Modemul folosește modulație în fază sau modulație în amplitudine?

24. Într-o diagramă-constelație, toate punctele sunt situate pe un cerc centrat în origine. Ce fel de modulație se folosește ?

25. Câte frecvențe folosește un modem full-duplex QAM-64?

26. Un sistem ADSL care folosește DMT alocă 3/4 dintre canalele de date disponibile pentru legătura de flux descendent. Se folosește o modulație QAM-64 pe fiecare canal. Care este capacitatea legăturii de flux descendent?

27. În exemplul de LMDS cu patru sectoare din fig. 2-30, fiecare sector are propriul său canal de 36 Mbps. Potrivit teoriei cozilor, dacă un canal este 50% plin, timpul de așteptare va fi egal cu timpul de transfer descendent. În aceste condiții, cât durează să se transfere o pagină Web de 5 KB ? Cât timp durează să se transfere aceeași pagină pe o linie ADSL de 1 Mbps ? Dar cu un modem de 56 Kbps ?

28. Zece semnale, fiecare necesitând 4000 Hz, sunt multiplexate în același canal utilizând FDM. Care este lățimea de bandă minimă necesară pentru canalul multiplexat? Presupunem că benzile suplimentare (de gardă) au lățimea de 400 Hz.

29. De ce a fost stabilit timpul de eșantionare PCM la 125 μsec?

30. Care este procentul de supraîncărcare pe o purtătoare T1? Mai exact, ce procent din cei 1.544 Mbps nu este pus la dispoziția utilizatorului final?

31. Comparați rata maximă de transfer al datelor pe un canal de 4 KHz, fără zgomot, care folosește:
a) codificare analogică (de ex. QPSK) cu 2 biți pe eșantion;
b) sistemul PCM T1.

32. Dacă un sistem cu purtătoarea T1 se desincronizează și pierde tactul, el încearcă să se resincronizeze folosind primul bit din fiecare cadru. Câte cadre vor trebui inspectate, în medie, pentru a se resincroniza cu o probabilitate de eșec de 0,001?

33. Care este diferența, dacă există vreuna, între blocul de demodulare a unui modem și blocul de codificare a unui codec? (În definitiv, ambele convertesc semnale analogice în semnale digitale.)

34. Un semnal este transmis digital pe un canal de 4 KHz, fără zgomot, cu un eșantion la fiecare 125 μs. Câți biți sunt de fapt transmiși pe secundă pentru fiecare dintre aceste metode de codificare:
a) Standardul CCITT de 2,048 Mbps;
b) DPCM cu o valoare relativă a semnalului pe 4 biți;
c) Modulația delta.

35. Un semnal pur sinusoidal de amplitudine A este codificat folosind modulația delta, cu x eșantioane/secundă. Un semnal de ieșire de +1 corespunde unei schimbări a semnalului cu $+A/8$ iar un semnal de ieșire de -1 corespunde unei schimbări a semnalului cu $-A/8$. Care este cea mai mare frecvență care poate fi urmărită fără erori cumulative?

36. Ceasurile SONET au o rată de deviație de aproximativ $1/10^9$. Cât timp este necesar pentru ca deviația să egaleze lățimea unui bit? Care sunt implicațiile acestui calcul?

37. În fig. 2-37 rata de transfer a datelor utilizatorului pentru OC-3 a fost stabilită la 148,608 Mbps. Arătați cum poate fi obținut acest număr din parametrii SONET OC-3.

38. Pentru a fi integra rate de transmisie a datelor mai mici decât STS-1, SONET are un sistem de fluxuri parțiale virtuale (VT, eng. Virtual Tributaries). O VT este o încărcătură utilă parțială care poate fi inserată într-un cadru STS-1 și combinată cu alte încărcături parțiale pentru a completa un cadru de date. VT1.5 folosește 3 coloane, VT2 folosește 4 coloane, VT3 folosește 6 coloane, iar VT6 folosește 12 coloane dintr-un cadru STS-1. Care dintre VT poate integra:
a) un serviciu DS-1 (1.544 Mbps) ?
b) serviciul european CEPT-1 (2.048 Mbps) ?
c) un serviciu DS-2 (6.312 Mbps) ?

39. Care este diferența esențială dintre comutarea de mesaje și comutarea de pachete ?

40. Care este lățimea de bandă disponibilă utilizatorului într-o conexiune OC-12c?

41. Se dau trei rețele cu comutare de pachete, conținând fiecare câte n noduri. Prima rețea are o topologie stea cu un comutator central, cea de-a doua este un inel (bidirecțional), iar cea de-a treia este interconectată complet, având câte o legătură de la fiecare nod către toate celelalte noduri. Care sunt lungimile căilor, măsurate în salturi (între noduri), pentru cazul cel mai bun, pentru cazul mediu și pentru cazul cel mai defavorabil?

42. Comparați întârzierea în transmisia unui mesaj de x biți pe o cale de k salturi (între noduri) dintr-o rețea cu circuite comutate și într-o rețea cu comutare de pachete (puțin aglomerată). Timpul de stabilire a circuitului este S sec, întârzierea de propagare este d sec/salt, dimensiunea pachetului este de p biți și rata de transfer a datelor este b biți/sec. În ce condiții rețeaua cu comutare de pachete are o întârziere mai mică?

43. Presupunem că x biți de date ale utile trebuie transmiși pe o cale de k salturi (între noduri), într-o rețea cu comutare de pachete, ca o serie de pachete; fiecare pachet conține p biți de date și h biți pentru antet, cu $x >> (p+h)$. Rata de transfer a liniei este de b bps și întârzierea de propagare este neglijabilă. Ce valoare a lui p minimizează întârzierea totală?

44. Într-un sistem tipic de telefonie mobilă cu celule hexagonale, este interzis să se refolosească banda de frecvențe a unei celule într-o celulă adiacentă. Dacă sunt disponibile în total de 840 frecvențe, câte frecvențe se pot folosi într-o anumită celulă?

45. Aranjamentul real a celulelor este rareori atât de regulat precum cel din fig. 2-41. Până și formele celulelor individuale sunt de obicei neregulate. Dați un motiv plauzibil pentru acest fapt.

46. Faceți o estimare sumară a numărului de microcelule PCS de 100 m diametru, care ar fi necesare pentru a acoperi San Francisco (120 km^2).

47. Uneori, atunci când un utilizator traversează granița dintre două celule, apelul curent se termină brusc, deși toate emițătoarele și receptoarele funcționează perfect. De ce?

48. D-AMPS oferă o calitate a vocii sensibil mai slabă decât GSM. Este adevărat că acest fapt se datorează cerinței ca D-AMPS să păstreze compatibilitatea cu AMPS, în timp ce GSM nu a avut astfel de constrângeri? Dacă nu, care este cauza ?

49. Calculați numărul maxim de utilizatori pe care D-AMPS îi poate suporta simultan într-o singură celulă. Faceți același calcul pentru GSM. Explicați diferența.

50. Să presupunem că A, B și C transmit simultan biți 0 folosind sistemul CDMA cu secvențele de felii din fig. 2-45(b). Care este secvența de felii rezultată?

51. În discuția despre ortogonalitatea secvențelor de felii CDMA, s-a afirmat că dacă $\mathbf{S} \bullet \mathbf{T} = 0$ atunci și $\mathbf{S} \bullet \overline{\mathbf{T}} = 0$. Demonstrați acest fapt.

52. Considerați un alt mod de a privi proprietatea de ortogonalitate a secvențelor de felii CDMA: fiecare bit dintr-o pereche de secvențe se poate potrivi sau nu. Exprimați proprietatea de ortogonalitate în termeni de potriviri și nepotriviri.

53. Un receptor CDMA primește următoarele felii : (-1 +1 –3 +1 -1 -3 +1 +1). Folosind secvențele de felii definite în fig. 2-45 (b), care dintre stații au transmis și ce biți a transmis fiecare?

54. La nivelul inferior, sistemul telefonic este construit în formă de stea, cu toate buclele locale dintr-un cartier convergente către un oficiu final. Din contră, televiziunea prin cablu este alcătuită dintr-un singur cablu lung, cu un traseu șerpuit pe deasupra tuturor caselor din același cartier. Presupunem că în viitor cablul TV va fi din fibră optică de 10 Gbps în loc de cupru. Ar putea acesta fi folosit pentru a simula modelul telefonic în care fiecare să aibă propria sa linie către oficiul final? Dacă da, câte case cu un telefon pot fi conectate la o singură fibră optică?

55. Un sistem de TV prin cablu are 100 de canale comerciale, fiecare din acestea alternând programele cu publicitatea. Această organizare seamănă cu TDM sau cu FDM ?

56. O companie de cablu decide să ofere acces Internet prin cablu într-un cartier cu 5000 de case. Compania folosește un cablu coaxial și o alocare de spectru care oferă o lățime de bandă de 100 Mbps pentru flux descendent pentru fiecare cablu. Pentru a atrage clienții, compania decide să garanteze cel puțin 2 Mbps lățime de bandă pentru flux descendent pentru fiecare casă, în orice moment. Descrieți modul în care trebuie să acționeze compania de cablu pentru a oferi această garanție.

57. Utilizând alocarea spectrală ilustrată în fig. 2-48 și informațiile date în text, câți Mbps alocă un sistem de cablu pentru fluxul ascendent și câți pentru cel descendent ?

58. Cât de repede poate un utilizator de cablu să primească date dacă rețeaua este în rest inactivă ?

59. Multiplexarea fluxurilor multiple de date STS-1, numite fluxuri parțiale (eng: Tributaries), joacă un rol important în SONET. Un multiplexor 3:1 multiplexează trei fluxuri parțiale STS-1 primi-

te la intrare într-un flux de ieșire STS-3. Multiplexarea este făcută octet cu octet, adică primii trei octeți de ieșire sunt primii octeți ai fluxurilor parțiale 1, 2 respectiv 3. Următorii octeți de ieșire alcătuiesc al doilea grup de fluxuri 1, 2 și 3, și așa mai departe. Scrieți un program care să simuleze acest multiplexor 3:1. Programul va conține 5 procese. Procesul principal creează patru alte procese, câte unul pentru fiecare dintre cele 3 fluxuri parțiale STS-1 și unul pentru multiplexor. Fiecare proces de tip flux parțial citește un cadru STS-1 din fișierul de intrare ca pe o succesiune de 810 biți și își trimite cadrul octet cu octet procesului multiplexor. Procesul multiplexor recepționează octeții și furnizează la ieșire un cadru STS-3 (tot octet cu octet) prin afișarea pe ecran. Pentru comunicația între procese folosiți conducte (eng: pipe).

3

NIVELUL LEGĂTURĂ DE DATE

În acest capitol vom studia arhitectura nivelului 2, nivelul legătură de date. Acest studiu se ocupă de algoritmii de obținere a unei comunicații eficiente și sigure, între două mașini adiacente la nivelul legăturii de date. Prin adiacență înțelegem că cele două mașini sunt conectate fizic printr-un canal de comunicație care se manifestă conceptual ca un fir (de exemplu, un cablu coaxial, o linie telefonică sau un canal de comunicație fără fir, de tip punct la punct). Calitatea esențială a unui canal care îl face asemănător unui fir este aceea că biții sunt livrați în exact aceeași ordine în care sunt transmiși.

La început ați putea crede că această problemă este atât de simplă, încât nu există programe de analizat - mașina A pune biții pe fir și mașina B îi preia. Din păcate, circuitele de comunicație produc uneori erori. În plus, ele au numai o rată finită a datelor și există o întârziere a propagării, nenulă, între momentul în care un bit este emis și momentul în care acesta este recepționat. Aceste limitări au implicații importante pentru eficiența transferului de date. Protocoalele utilizate pentru comunicație trebuie să ia în considerare toți acești factori. Aceste protocoale reprezintă subiectul capitolului de față.

După o introducere în principalele aspecte ale proiectării nivelului legătură de date, vom începe studiul protocoalelor examinând natura erorilor, cauzele producerii lor și cum pot fi ele detectate și corectate. Apoi vom studia o serie de protocoale din ce în ce mai complexe, fiecare dintre ele rezolvând cât mai multe dintre problemele prezente la acest nivel. Vom încheia cu un studiu al modelării și corectitudinii protocoalelor și vom da câteva exemple de protocoale ale legăturii de date.

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE

Nivelul legătură de date are un număr de funcții specifice pe care trebuie să le îndeplinească. Aceste funcții includ:

1. Furnizarea unei interfețe bine-definite către nivelul rețea
2. Tratarea erorilor de transmisie
3. Reglarea fluxului cadrelor în așa fel, încât receptorii lenți să nu fie inundați de către emițători rapizi.

Pentru a îndeplini aceste scopuri, nivelul legătură de date primește pachete de la nivelul rețea, pe care le încapsulează în **cadre** în vederea transmiterii. Fiecare cadru conține un antet, un câmp de informație utilă pentru pachet și încheiere, după cum se vede în fig. 3-1. Gestionarea cadrelor reprezintă esența a ceea ce face nivelul legătură de date. In secțiunile următoare vom examina in detaliu toate aspectele menționate anterior.

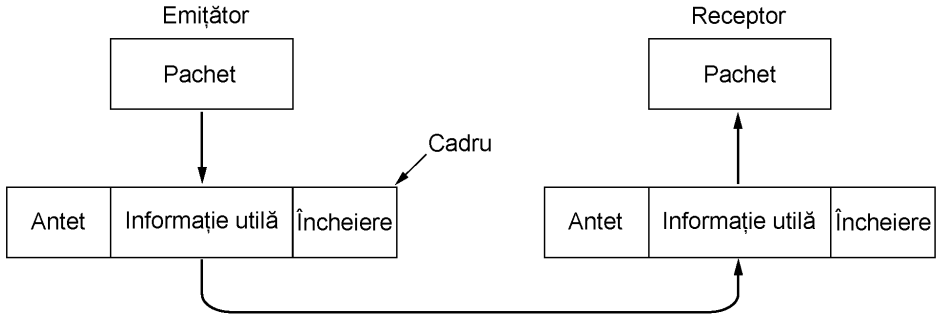


Fig. 3-1. Relația dintre pachete și cadre.

Cu toate că acest capitol se referă numai la nivelul legătură de date și la protocoalele legăturii de date, multe dintre principiile pe care le vom studia aici, cum ar fi controlul erorilor și controlul fluxului, se regăsesc și în protocoalele de transport, și în alte protocoale. În realitate, în multe rețele, aceste funcții se găsesc doar la nivelurile superioare, nu și la nivel legătură de date. Oricum, indiferent de nivelul la care se găsesc, principiile sunt aproximativ aceleași, deci nu contează prea mult unde le studiem. La nivelul legăturii de date ele apar de obicei în forma cea mai simplă și cea mai pură, făcând din acest nivel un loc foarte potrivit studierii detaliate a acestor principii.

3.1.1 Servicii oferite nivelului rețea

Funcția nivelului legătură de date este să ofere servicii nivelului rețea. Principalul serviciu este transferul datelor de la nivelul rețea al mașinii sursă la nivelul rețea al mașinii destinație. La nivelul rețea al mașinii sursă există o entitate, să-i spunem proces, care trimite biți către nivelul legătură de date, pentru a fi transmiși la destinație. Funcția nivelului legătură de date este să transmită biții spre mașina destinație, pentru ca acolo să fie livrați nivelului rețea, așa cum se arată în fig. 3-2(a). Transmisia efectivă urmează calea din fig. 3-2(b), dar este mai ușor de înțeles în termenii a două procese

ale nivelului legătură de date care comunică utilizând un protocol al legăturii de date. Din acest motiv, pe parcursul acestui capitol, vom folosi în mod implicit modelul din fig. 3-1(a).

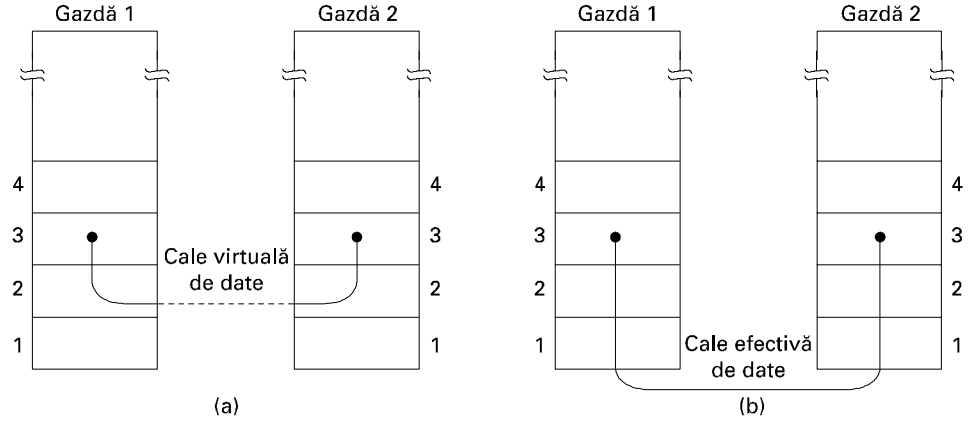


Fig. 3-2. (a) Comunicație virtuală. (b) Comunicație efectivă.

Nivelul legătură de date poate fi proiectat să ofere diferite servicii. Serviciile efective oferite pot varia de la sistem la sistem. Trei posibilități de bază, oferite în mod curent, sunt:

1. Serviciu neconfirmat fără conexiune.
2. Serviciu confirmat fără conexiune.
3. Serviciu confirmat orientat-conexiune.

Să le analizăm pe rând pe fiecare dintre acestea.

Serviciul neconfirmat fără conexiune constă din aceea că mașina sursă trimite cadre independente către mașina destinație, fără ca mașina destinație să trebuiască să confirme primirea lor. În acest caz, nu sunt necesare stabilirea și desființarea unei conexiuni logice. Dacă un cadru este pierdut datorită zgomotului de pe linie, la nivelul legătură de date nu se face nici o încercare pentru recuperarea lui. Această clasă de servicii este adecvată atunci când rata de erori este foarte scăzută, încât recuperarea este lăsată în sarcina nivelurilor superioare. De asemenea, este adecvată pentru traficul de timp real, cum ar fi cel de voce, unde a primi date cu întârziere este mai rău decât a primi date eronate. Majoritatea LAN-urilor utilizează la nivelul legăturii de date servicii neconfirmate fără conexiune.

Următorul pas în ceea ce privește siguranța este serviciul confirmat fără conexiune. Atunci când este oferit acest serviciu, încă nu se utilizează conexiuni, dar fiecare cadru trimis este confirmat individual. În acest mod, emițătorul știe dacă un cadru a ajuns sau nu cu bine. Dacă nu a ajuns într-un interval de timp specificat, poate fi trimis din nou. Acest serviciu este folositor pentru canale nesigure, cum ar fi sistemele fără fir.

Poate că merită să subliniem că asigurarea confirmării la nivelul legăturii de date este doar o optimizare, niciodată o cerință. Nivelul rețea poate întotdeauna să trimită un mesaj și să aștepte să fie confirmat. Dacă confirmarea nu apare în timp util, atunci emițătorul poate retrimite întregul mesaj.

Problema cu această strategie este aceea că, de obicei, cadrele au o lungime maximă impusă de hardware, iar pachetele nivelului rețea nu au aceasta limitare. Dacă pachetul mediu este spart în, să zicem, 10 cadre și 20% din totalul cadrelor sunt pierdute, transmiterea acestuia poate lua foarte mult timp. În cazul în care cadrele individuale sunt confirmate și retransmise, pachetele întregi vor fi transmise mult mai rapid. Pe canale sigure, precum fibra optică, costul suplimentar implicat de un

astfel de protocol al legăturii de date poate fi nejustificat, dar pe canale fără fir, costul este pe deplin justificat datorită nesiguranței acestora.

Revenind la serviciile noastre, cel mai sofisticat serviciu pe care nivelul legătură de date îl pune la dispoziția nivelului rețea este serviciul orientat-conexiune. În cazul acestui serviciu, mașinile sursă și destinație stabilesc o conexiune înainte de a transfera date. Fiecare cadru trimis pe conexiune este numerotat și nivelul legătură de date garantează că fiecare cadru trimis este într-adevăr recepționat. Mai mult, garantează că fiecare cadru este recepționat exact o dată și toate cadrele sunt recepționate în ordinea corectă. În schimb, în cazul serviciului fără conexiune, este posibil ca, datorită unei confirmări pierdute, un cadru să fie transmis de mai multe ori și, prin urmare, recepționat de mai multe ori. Spre deosebire de acesta, serviciul orientat conexiune furnizează proceselor de la nivelul rețea echivalentul unui flux de biți sigur.

Atunci când este utilizat serviciul orientat conexiune, transferurile au trei faze distincte. În prima fază este stabilită conexiunea, ambele părți inițializând variabile și contoare, utilizate pentru a ține evidența cadrelor care au fost recepționate și a celor care nu au fost. În a doua fază, sunt transmise unul sau mai multe cadre. În a treia și ultima fază, conexiunea este desființată, eliberând variabilele, tampoanele și alte resurse utilizate la menținerea conexiunii.

Să considerăm un exemplu tipic: o subrețea WAN formată din rutere conectate prin linii telefonice punct-la-punct, închiriate. Când un cadru ajunge la un ruter, hardware-ul verifică absența erorilor (folosind tehnici pe care le vom studia mai târziu în acest capitol), și trimite cadrul programelor nivelului legătură de date (care se pot afla într-un cip de pe adaptorul de rețea). Programele nivelului legătură de date verifică dacă acesta este cadrul așteptat și, dacă este așa, trimit pachetul din câmpul de informație utilă către programele de dirijare. Programele de dirijare aleg linia de ieșire corespunzătoare și trimit pachetul înapoi, către programele nivelului legătură de date, care apoi îl transmit. Fluxul dintre două rutere este reprezentat în fig. 3-3.

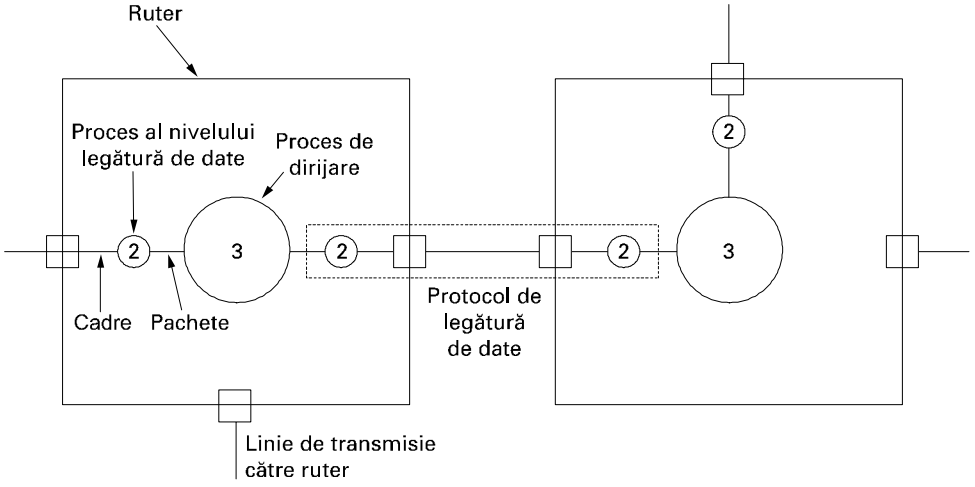


Fig. 3-3. Plasarea protocolului legătură de date.

Programul de dirijare dorește de obicei ca operația să fie corect executată, ceea ce presupune conexiuni secvențiale sigure pe fiecare linie punct-la-punct. El nu vrea să fie deranjat prea des de pachete care s-au pierdut pe drum. Este sarcina protocolului legăturii de date, prezentat în dreptun-

ghiul punctat, să facă liniile de comunicație nesigure să pară perfecte sau, cel puțin, suficient de bune. Această proprietate este foarte importantă pentru legăturile fără fir, care sunt, prin natura lor, foarte nesigure. Ca o remarcă, cu toate că am prezentat copii ale programelor nivelului legătură de date în fiecare ruter, de fapt există o singură copie, care tratează toate liniile, utilizând tabele și structuri de date diferite pentru fiecare dintre ele.

3.1.2 Încadrarea

În vederea furnizării unui serviciu nivelului rețea, nivelul legătură de date trebuie să utilizeze serviciul furnizat de către nivelul fizic. Sarcina nivelului fizic este să primească un flux de biți și să încerce să-l trimită la destinație. Nu se garantează că acest flux de biți nu conține erori. Numărul de biți recepționați poate fi mai mic, egal cu, sau mai mare decât numărul de biți transmiși și pot avea valori diferite. Este la latitudinea nivelului legătură de date să detecteze și, dacă este necesar, să corecteze erorile.

Abordarea uzuală pentru nivelul legătură de date este să spargă șirul de biți în cadre discrete și să calculeze suma de control pentru fiecare cadru. (Algoritmii pentru suma de control vor fi discutați mai târziu în acest capitol.) Atunci când un cadru ajunge la destinație, suma de control este recalculată. Dacă noua sumă de control este diferită de cea conținută în cadru, nivelul legătură de date știe că a apărut o eroare și face operațiile necesare pentru a o rezolva (de exemplu, elimină cadrul eronat și probabil trimite înapoi un raport de eroare).

Spargerea șirului de biți în cadre este mai dificilă decât pare la prima vedere. O cale pentru a realiza această încadrare este inserarea de intervale de timp între cadre, așa cum inserăm spații între cuvinte într-un text normal. Totuși, rețelele dau rareori garanții referitoare la timp, așa că este posibil ca aceste intervale să fie comprimate sau ca în timpul transmisiei să fie inserate alte intervale.

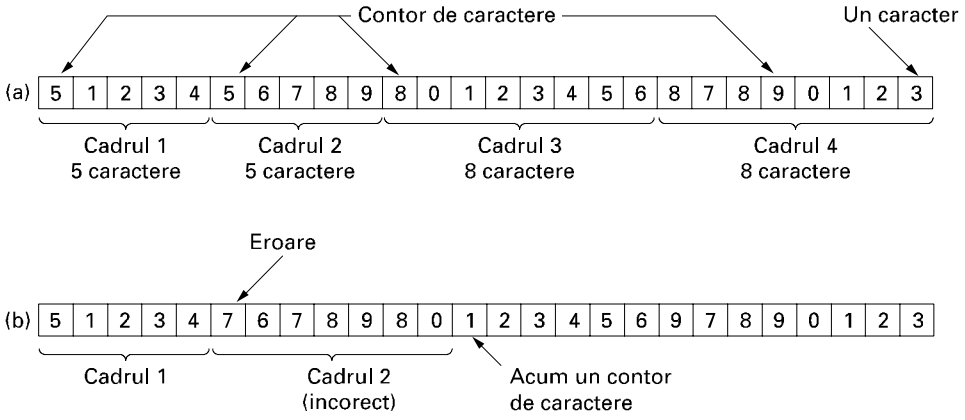

Contor de caractere
Un caracter
(a)
5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Cadrul 1
5 caractere
Cadrul 2
5 caractere
Cadrul 3
8 caractere
Cadrul 4
8 caractere
Eroare
(b)
5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 9 7 8 9 0 1 2 3
Cadrul 1
Cadrul 2
(incorect)
Acum un contor
de caractere

Fig. 3-4. Un șir de caractere. (a) Fără erori. (b) Cu o eroare.

Deoarece este prea periculos să ne bizuim pe timp pentru a marca începutul și sfârșitul fiecărui cadru, au fost elaborate alte metode. În această secțiune vom analiza patru metode:

1. Numărarea caracterelor.
2. Indicatori cu inserare de octeți.
3. Indicatori de început și de sfârșit, cu inserare de biți.
4. Violarea codificărilor la nivel fizic.

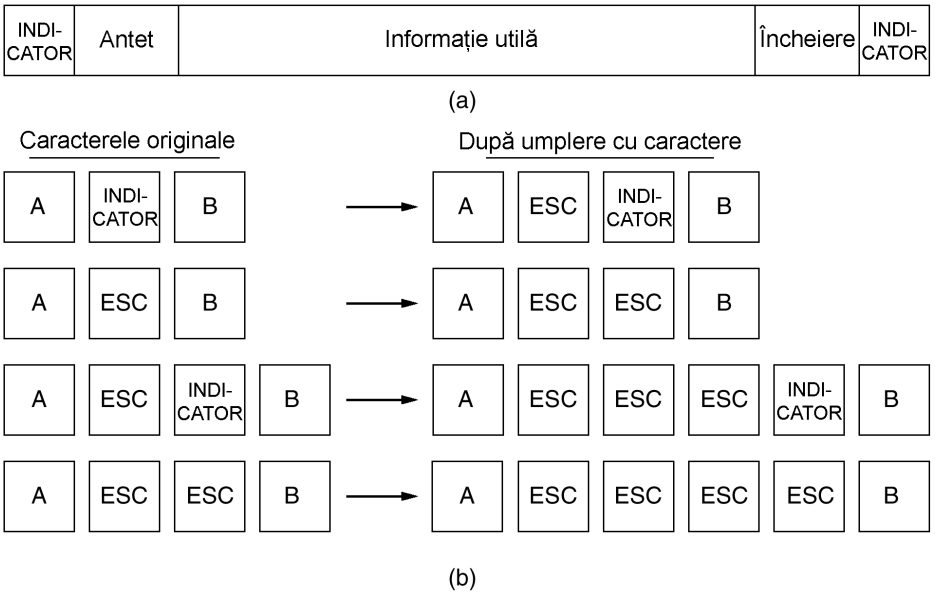


Fig. 3-5. (a) Cadru delimitat de octeți indicatori. (b) Patru exemple de secvențe de octeți înainte și după inserare

Prima metodă de încadrare utilizează un câmp din antet pentru a specifica numărul de caractere din cadru. Atunci când nivelul legătură de date de la destinație primește contorul de caractere, știe câte caractere urmează și unde este sfârșitul cadrului. Această tehnică este prezentată în fig. 3-4 (a) pentru patru cadre de dimensiune de 5, 5, 8 și 8 caractere.

Problema cu acest algoritm este că valoarea contorului poate fi alterată de erori de transmisie. De exemplu, dacă contorul de caractere din al doilea cadru din fig. 3-4(b) din 5 devine 7, destinația va pierde sincronizarea și va fi incapabilă să localizeze începutul cadrului următor. Chiar dacă suma de control este incorectă și destinația știe că a primit cadru eronat, nu există nici o posibilitate de a determina unde începe următorul cadru. Nu ajută nici trimiterea unui cadru înapoi la sursă, cerând o retransmisie, deoarece destinația nu știe peste câte caractere să sară pentru a începe retransmisia. Din acest motiv, metoda contorizării caracterelor este rar utilizată.

A doua metodă de încadrare înlătură problema resincronizării după o eroare, prin aceea că fiecare cadru începe și se termină cu o secvența specială de octeți. Inițial, octeții ce indicau începutul, respectiv sfârșitul erau diferiți, dar în ultimii ani s-a trecut la utilizarea unui singur octet, numit octet indicator, atât ca indicator de început, cât și de sfârșit, așa cum se prezintă in fig. 3-5(a). În acest fel, dacă receptorul pierde sincronizarea, acesta poate căuta octetul indicator pentru a găsi sfârșitul cadrului. Doi octeți indicatori consecutivi indică sfârșitul unui cadru și începutul celui care urmează.

O problemă serioasă cu această metodă apare atunci când se transmit date binare, cum ar fi un obiect sau numere în virgulă mobilă. Se poate întâmpla ca în date să apară octetul folosit ca indicator. Această situație interferează cu procesul de încadrare. O cale de rezolvare a acestei probleme este ca nivelul legătură de date al emițătorului să insereze un octet special (ESC) înaintea fiecărei apariții „accidentale” a indicatorului în date. Nivelul legătură de date al receptorului va elimina acest octet special înainte de a pasa datele nivelului rețea. Această tehnică poartă numele de **inserare de octeți** (eng.: **byte stuffing**) sau **inserare de caractere** (eng.: **character stuffing**) . Deci, un octet indica-

tor utilizat pentru încadrare poate fi diferențiat de unul prezent în date prin faptul că este sau nu precedat de un octet special.

Bineînțeles, următoarea întrebare este: Ce se întâmplă dacă un octet special apare în mijlocul datelor? Răspunsul este că pe lângă el se inserează încă un octet special.

Deci, un singur octet special face parte dintr-o secvență specială, iar un octet special dublu indica faptul ca un singur octet de acest tip a apărut in cadrul datelor. Câteva exemple sunt prezentate in fig. 3-5(b). În toate cazurile, secvența de octeți obținută după eliminarea octeților inserați este exact secvența originală.

Schema de inserare de octeți prezentată în fig. 3-5 reprezintă o ușoară simplificare a schemei utilizate în cadrul protocolului PPP, pe care majoritatea calculatoarelor casnice îl folosesc pentru a comunica cu furnizorul de servicii Internet. Vom discuta despre PPP mai târziu în acest capitol.

Un dezavantaj major al utilizării acestei metode de încadrare este acela că este limitată la utilizarea caracterelor de 8 biți. Nu toate codurile utilizează caractere de 8 biți. De exemplu, UNICODE folosește caractere pe 16 biți. Datorită dezvoltării rețelelor, dezavantajele inserării de coduri de caractere în mecanismul de încadrare au devenit din ce în ce mai evidente, așa că a trebuit dezvoltată o nouă tehnică, care să permită caractere de dimensiune variabilă.

Noua tehnică permite cadrelor de date să conțină un număr arbitrar de biți și permite coduri de caractere cu un număr arbitrar de biți per caracter. Funcționează astfel: fiecare cadru începe și se termină cu un șablon special pe biți, 01111110, numit octet **indicator (flag)**. De fiecare dată când nivelul legătură de date al emițătorului identifică cinci de unu consecutivi în date, inserează automat un bit 0 în șirul de biți de rezultați. Această **inserare de biți (bit stuffing)** este similară inserării de caractere, în care un octet escape este inserat în șirul de caractere de ieșire, înainte de fiecare octet indicator din date.

Atunci când receptorul primește o succesiune de cinci biți 1, urmați de un bit 0, extrage automat (adică șterge) bitul 0. La fel ca și inserarea de caractere, care este complet transparentă pentru nivelul rețea din ambele calculatoare, așa este și inserarea de biți. Dacă datele utilizator conțin șablonul indicator, 01111110, acest indicator este transmis ca 011111010, dar în memoria receptorului este păstrat ca 01111110. Fig. 3-6 dă un exemplu de inserare de biți.

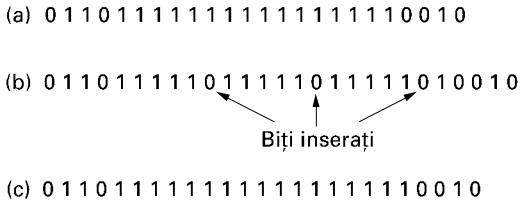


Fig. 3-6. Inserare de biți. (a) Datele originale. (b) Datele așa cum apar pe linie. (c) Datele așa cum sunt stocate în memoria receptorului după extragerea biților inserați.

În cazul inserării de biți, granițele dintre două cadre pot fi recunoscute fără ambiguitate datorită șablonului indicator. Astfel, dacă receptorul pierde evidența a ceea ce primește, tot ceea ce are de făcut este să caute la intrare secvențele indicator, deoarece acestea pot să apară numai la marginile cadrului și niciodată în interiorul datelor.

Ultima metodă de încadrare este aplicabilă rețelelor în care codificarea pe mediul fizic conține o anumită redundanță. De exemplu, unele LAN-uri codifică un bit de date utilizând doi biți fizici. De obicei, un bit 1 este reprezentat de o tranziție sus-jos și un bit 0 de o tranziție jos-sus. Aceasta înseamnă că fiecare bit de date are o tranziție în mijloc, receptorului fiindu-i ușor să localizeze frontie-

rele biților. Combinațiile sus-sus și jos-jos nu sunt utilizate pentru date, dar sunt utilizate pentru delimitarea cadrelor in unele protocoale.

Ca observație finală asupra încadrării, multe protocoale de legătură de date utilizează, pentru o mai mare siguranță, o combinație de contor de caractere cu una dintre celelalte metode. La sosirea unui cadru, pentru a localiza sfârșitul acestuia, este utilizat câmpul contor. Cadrul este acceptat ca valid doar dacă în poziția respectivă există delimitatorul corespunzător și dacă suma de control este corectă. Altfel, șirul de intrare este scanat pentru a detecta următorul delimitator.

3.1.3 Controlul erorilor

Problema marcării începutului și sfârșitului fiecărui cadru fiind rezolvată, ne vom îndrepta atenția către problema următoare: cum să ne asigurăm că toate cadrele ajung până la urmă la nivelul rețea de la destinație și în ordinea corectă. Să presupunem că emițătorul trimite cadrele de ieșire fără să verifice dacă au ajuns corect. Așa ceva se poate accepta în cazul serviciilor neconfirmate fără conexiune, dar nu în cel al serviciilor sigure, orientate conexiune.

Modul uzual de a asigura o transmitere sigură este de a furniza emițătorului o reacție inversă (eng.: feedback) despre ceea ce se întâmplă la celălalt capăt al liniei. De obicei protocolul îi cere receptorului să trimită înapoi cadre de control speciale, purtând confirmări pozitive sau negative despre cadrele sosite. Dacă emițătorul recepționează o confirmare pozitivă despre un cadru, el știe că acel cadru a ajuns cu bine. Pe de altă parte, o confirmare negativă înseamnă că ceva a mers prost și cadrul trebuie retransmis.

O complicație în plus vine de la posibilitatea ca defectele de echipament să determine dispariția completă a unui cadru (de exemplu într-o rafală de zgomot). În acest caz, receptorul nu va reacționa în nici un fel, din moment ce nu are nici un motiv să reacționeze. Trebuie să fie clar că un protocol în care emițătorul trimite un cadru și apoi așteaptă o confirmare, pozitivă sau negativă, va rămâne agățat pentru totdeauna dacă un cadru este complet pierdut datorită, de exemplu, nefuncționării echipamentului.

Această posibilitate a determinat introducerea contoarelor de timp (ceasurilor) la nivelul legăturii de date. Atunci când emițătorul trimite un cadru, pornește de obicei și un contor de timp. Contorul de timp este setat să expire după un interval suficient de lung pentru ca acel cadru să poată ajunge la destinație, să fie prelucrat acolo și confirmarea să se propage înapoi către emițător. În mod normal, cadrul va fi corect recepționat și confirmarea va sosi înainte ca timpul să expire, caz în care contorul va fi anulat.

Dar, dacă fie cadrul, fie confirmarea se pierd, intervalul de timp expiră, în acest caz, emițătorul fiind atenționat că a apărut o problemă. Soluția evidentă este retransmiterea cadrului. Dar, atunci când cadrele pot fi transmise de mai multe ori, există pericolul ca receptorul să accepte același cadru de două sau mai multe ori și să-l trimită de mai multe ori nivelului rețea. Pentru a evita această situație este necesar să atribuim numere de secvență cadrelor de ieșire, așa încât receptorul să poată face distincție între cadrele retransmise și cele originale.

Întreaga problematică a gestiunii ceasurilor și numerelor de secvență, astfel încât fiecare cadru să ajungă la nivelul rețea de la destinație o singură dată nici mai mult, nici mai puțin, reprezintă o parte importantă a obligațiilor nivelului legătură de date. Mai târziu, în acest capitol, urmărind o serie de exemple de complexitate din ce în ce mai mare, vom studia în detaliu cum este realizată această gestiune.

3.1.4 Controlul fluxului

Un alt aspect important de proiectare care apare la nivelul legătură de date (și, desigur, și la nivelurile superioare) este cum trebuie procedat cu un emițător care dorește în mod sistematic să transmită cadre mai repede decât poate să accepte receptorul. Această situație poate să apară ușor atunci când emițătorul rulează pe un calculator rapid (sau mai puțin încărcat) și receptorul rulează pe o mașină lentă (sau foarte încărcată). Emițătorul continuă să transmită cadre la o rată înaltă până când receptorul este complet inundat. Chiar dacă transmisia este fără erori, la un anumit punct receptorul nu va mai fi capabil să trateze cadrele care sosesc și va începe să piardă unele dintre ele. Bineînțeles, trebuie făcut ceva pentru a evita această situație.

Există două abordări des utilizate. In cazul celei dintâi, **controlul fluxului bazat pe reacție (feedback-based flow control)**, receptorul acordă emițătorului permisiunea de a mai transmite date, sau cel puțin comunică emițătorului informații despre starea sa. În cea de-a doua, **controlul fluxului bazat pe rată (rate-based flow control)**, protocolul dispune de un mecanism integrat care limitează rata la care emițătorul poate transmite, fără a folosi informații de la receptor. În acest capitol vom studia scheme de control al fluxului bazat pe reacție, deoarece la nivelul legătură de date nu se utilizează controlul fluxului bazat pe rată. Vom studia acest tip de control al fluxului în cap. 5.

Sunt cunoscute diferite scheme de control al fluxului, dar cele mai multe dintre ele utilizează același principiu de bază. Protocolul conține reguli bine definite despre momentul când emițătorul poate trimite următorul cadru. Deseori aceste reguli interzic trimiterea cadrelor înainte ca receptorul să o permită, implicit sau explicit. De exemplu, când se stabilește o conexiune, receptorul trebuie să spună: "Acum poți să-mi trimiți *n* cadre, dar după ce au fost trimise, să nu trimiți altele până când nu îți spun să continui". Vom examina detaliile în cele ce urmează.

3.2 DETECTAREA ȘI CORECTAREA ERORILOR

Așa cum am văzut în Cap. 2, sistemul telefonic are trei părți: comutatoarele, trunchiurile interoficii (eng.: interoffice trunks) și buclele locale. Primele două sunt acum aproape complet digitizate în majoritatea țărilor dezvoltate. Buclele locale sunt încă din perechi de fire torsadate din cupru și vor continua să fie așa ani întregi, din cauza costului enorm al înlocuirii lor . În timp ce pe partea digitală erorile sunt rare, ele sunt încă obișnuite pe buclele locale. Mai mult, comunicația fără fir a devenit mai uzuală și ratele erorilor sunt, în acest caz, cu câteva ordine de mărime mai proaste decât pe trunchiurile de fibră inter-oficii. Concluzia este: erorile de transmisie vor fi o realitate pentru mulți ani de acum înainte. Trebuie găsită o metodă de tratare a acestor erori.

Ca rezultat al proceselor fizice care le generează, erorile din unele medii (de exemplu radio) tind să vină mai curând în rafale decât izolate. Sosirea erorilor în rafală are atât avantaje cât și dezavantaje față de erorile izolate, de un singur bit. Avantajul este acela că datele de la calculator sunt trimise întotdeauna în blocuri de biți. Să presupunem că dimensiunea unui bloc este de 1000 de biți și rata de eroare este de 0.001 per bit. Dacă erorile ar fi independente, multe blocuri ar conține o eroare. Dacă erorile vin în rafală de câte 100, în medie vor fi afectate doar unul sau două blocuri din 100.

Dezavantajul erorilor în rafală este acela că sunt mult mai greu de detectat și corectat decât erorile izolate.

3.2.1 Coduri corectoare de erori

Proiectanții de rețele au dezvoltat două strategii de bază pentru tratarea erorilor. O modalitate este ca pe lângă fiecare bloc de date trimis să se includă suficientă informație redundantă pentru ca receptorul să poată deduce care a fost caracterul transmis. O altă soluție este să se includă suficientă redundanță pentru a permite receptorului să constate că a apărut o eroare, dar nu care este eroarea, și să ceară o retransmisie. Prima strategie utilizează **coduri corectoare de erori**, iar cea de-a doua utilizează **coduri detectoare de erori**. Folosirea codurilor corectoare de erori este deseori referită sub numele de **corectare de erori în avans (eng.: forward error correction)**.

Fiecare dintre aceste tehnici se utilizează in situații diferite. Pe canale cu siguranță mare, cum ar fi fibra optică, este mai eficient să utilizăm un cod detector de erori și să retransmitem blocul în care s-au detectat erori. In cazul canalelor de comunicație fără fir, este indicat sa adăugăm destulă informație redundantă fiecărui bloc, în loc să ne bazăm pe retransmisie, care poate să fie la rândul său afectată de erori.

Pentru a înțelege cum pot fi tratate erorile, este necesar să privim cu atenție la ceea ce este de fapt o eroare. În mod normal, un cadru conține m biți de date (adică mesaj) și r biți redundanți sau de control. Să considerăm lungimea totală n (adică, $n = m + r$). O unitate formată din n biți, care conține date și biți de control, este numită frecvent **cuvânt de cod** de n biți (eng.: n-bit **codeword**).

Date fiind două cuvinte de cod, să zicem, 10001001 și 10110001, este posibil să determinăm câți biți corespunzători diferă. În acest caz diferă 3 biți. Pentru a determina câți biți diferă, aplicăm operatorul SAU EXCLUSIV între cele două cuvinte de cod și numărăm biții 1 din rezultat, de exemplu:

```
10001001
10110001
00111000
```

Numărul de poziții binare în care două cuvinte de cod diferă se numește **distanța Hamming** (Hamming, 1950). Semnificația sa este că dacă două cuvinte de cod sunt despărțite de o distanță Hamming d, sunt necesare d erori de un singur bit pentru a-l converti pe unul în celălalt.

În multe aplicații de transmisie de date, toate cele 2^m mesaje de date posibile sunt corecte, dar, datorită modului în care sunt calculați biții de control, nu sunt utilizate toate cele 2^n cuvinte de cod posibile. Dat fiind algoritmul pentru calculul biților de control, este posibil să construim o listă completă de cuvinte de cod permise și din această listă să găsim cele două cuvinte de cod a căror distanță Hamming este minimă. Această distanță este distanța Hamming a codului complet.

Proprietățile detectoare și corectoare de erori ale unui cod depind de distanța sa Hamming. Pentru a detecta d erori, este nevoie de un cod cu distanță $d + 1$, deoarece cu un asemenea cod nu există nici o modalitate ca d erori de un singur bit să poată modifica un cuvânt de cod corect într-un alt cuvânt de cod corect. Atunci când receptorul vede un cuvânt de cod incorect, poate spune că s-a produs o eroare de transmisie. Similar, pentru a corecta d erori, este nevoie de un cod cu distanță $2d + 1$, deoarece în acest mod cuvintele de cod corecte sunt atât de distanțate, încât, chiar cu d modificări, cuvântul de cod originar este totuși mai apropiat decât alte cuvinte de cod și va fi unic determinat.

Ca un exemplu simplu de cod detector de erori, să considerăm un cod în care la date este adăugat un singur bit de paritate. Bitul de paritate este ales astfel, încât numărul de biți 1 din cuvântul de cod să fie par (sau impar). De exemplu, atunci când 1011010 este trimis în paritate pară, prin adăugarea unui bit la sfârșit devine 10110100. Cu paritatea impară , 1011010 devine 10110101. Un cod cu

un singur bit de paritate are distanța 2, deoarece orice eroare pe un singur bit produce un cuvânt de cod cu paritatea greșită. Acesta poate fi utilizat pentru detectarea erorilor singulare.

Ca exemplu simplu de cod corector de erori, să considerăm un cod cu numai patru cuvinte de cod corecte:

0000000000, 0000011111, 1111100000 și 1111111111.

Acest cod are distanța 5, ceea ce înseamnă că poate corecta erori duble. Dacă sosește cuvântul de cod 0000000111, cel ce recepționează știe că originalul trebuie să fi fost 00000111111. Dacă totuși o eroare triplă modifică 0000000000 în 0000000111, eroarea nu va fi corectată corespunzător.

Să ne imaginăm că dorim să proiectăm un cod cu m biți de mesaj și r biți de control care ne va permite să corectăm toate erorile singulare. Pentru fiecare din cele 2^m mesaje corecte există n cuvinte de cod eronate, aflate la distanță 1 de el. Acestea sunt formate prin inversarea sistematică a fiecăruia dintre cei n biți din cuvântul de cod de n biți format din el. Astfel, fiecare din cele 2^m mesaje corecte necesită $n+1$ șabloane asociate. Cum numărul total de șabloane este 2^n, trebuie să avem $(n+1)\ 2^m <= 2^n$. Utilizând $n=m+r$, această condiție devine $(m + r + 1) <= 2^r$. Dându-se m, acesta impune o limită inferioară asupra numărului de biți de control necesari pentru a corecta erorile singulare.

Această limită inferioară teoretică poate fi, de fapt, atinsă utilizând o metodă atribuită lui Hamming (1950). Biții cuvântului de cod sunt numerotați consecutiv, începând cu bitul 1 de la marginea din stânga, bitul 2 imediat la dreapta sa, etc. Biții care sunt puteri ale lui 2 (1, 2, 4, 8, 16 etc.) sunt biți de control. Restul (3, 5, 6, 7, 9 etc.) sunt completați cu cei m biți de date. Fiecare bit de control forțează ca paritatea unui grup de biți, inclusiv el însuși, să fie pară (sau impară).

Un bit poate fi inclus în mai multe calcule de paritate. Pentru a vedea la care biți de control contribuie bitul de date din poziția k, rescriem k ca o sumă de puteri ale lui 2. De exemplu, 11= 1+2+8 și 29=1+4+8+16. Un bit este verificat de acei biți de control care apar în dezvoltarea sa (de exemplu, bitul 11 este verificat de biții 1, 2 și 8).

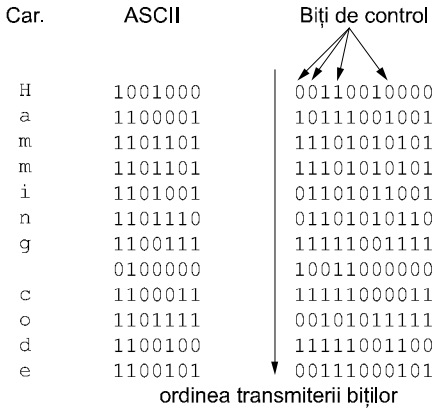


Fig. 3-7. Utilizarea unui cod Hamming pentru corectarea erorilor în rafală.

Când sosește un cuvânt de cod, receptorul inițializează un contor la 0. Acesta examinează apoi fiecare bit de control, k (k = 1, 2, 4, 8...) pentru a vedea dacă are paritatea corectă. Dacă nu, adaugă k la contor. Dacă, după ce au fost examinați toți biții de control, contorul este 0 (adică, dacă toți biții au fost corecți), cuvântul de cod este acceptat ca valid. Dacă valoarea contorului este nenulă, ea reprezintă numărul bitului incorect. De exemplu, dacă biții de control 1, 2 și 8 sunt eronați, atunci bitul inversat este 11, deoarece este singurul verificat de biții 1, 2 și 8. fig. 3-6 prezintă câteva caractere

ASCII pe 7 biți codificate prin cuvinte de cod pe 11 biți, utilizând codul Hamming. De reamintit că informația este regăsită în biții de pe pozițiile 3, 5, 6, 7, 9, 10 și 11. Codurile Hamming pot corecta numai erori singulare. Totuși, există un artificiu care poate fi utilizat pentru a permite codurilor Hamming să corecteze erorile în rafală. O secvență de k cuvinte de cod consecutive este aranjată ca o matrice, având câte un cuvânt de cod pe fiecare linie. În mod normal, datele ar fi transmise linie cu linie, de la stânga la dreapta. Pentru a corecta erorile în rafală, datele vor trebui transmise pe coloane, începând cu coloana cea mai din stânga. Când au fost trimiși toți cei k biți, este transmisă a doua coloană și așa mai departe, așa cum se arată in fig. 3-7. Atunci când un cadru ajunge la receptor, matricea este reconstruită, coloană cu coloană. Dacă a apărut o eroare în rafală, de lungime k, va fi afectat cel mult un bit din fiecare dintre cele k cuvinte de cod, dar codul Hamming poate corecta o eroare pe cuvânt de cod, așa încât întregul bloc poate fi refăcut. Această metodă utilizează kr biți de control pentru a face blocuri de km biți de date imune la erorile în rafală de lungime k sau mai mică.

3.2.2 Coduri detectoare de erori

Codurile corectoare de erori sunt adesea utilizate pe canale fără fir, care sunt cunoscute ca fiind predispuse la erori, în comparație cu firele de cupru sau fibra optică. Fără coduri detectoare de erori, comunicația ar fi greu de realizat. In cazul firelor de cupru sau a fibrei optice rata erorilor este mult mai mică, așa că detectarea erorilor și retransmisia este de obicei mai eficientă aici ca metodă de tratare a erorilor care apar ocazional.

Ca un exemplu simplu, să considerăm un canal în care erorile sunt izolate și rata erorilor este de 10^{-6} per bit. Să considerăm că dimensiunea unui bloc este de 1000 biți. Pentru a permite corecția erorilor pentru blocuri de 1000 de biți sunt necesari 10 biți de control; un megabit de date va necesita 10000 biți de control. Pentru a detecta ușor un bloc cu o singură eroare de un bit, va fi suficient un bit de paritate la fiecare bloc. O dată la fiecare 1000 de blocuri va trebui transmis un extra-bloc (1001 biți). Încărcarea suplimentară totală în cazul metodei de detecție și retransmisie este de numai 2001 biți pentru un megabit de date, în comparație cu 10000 biți pentru un cod Hamming.

Dacă unui bloc i se adaugă un singur bit de paritate și blocul este puternic deformat de o eroare în rafală lungă, probabilitatea ca eroarea să fie detectată este de numai 0.5, ceea ce este greu de acceptat. Șansele pot fi îmbunătățite considerabil dacă fiecare bloc transmis este privit ca o matrice dreptunghiulară de n biți lățime și k biți înălțime, după cum am arătat mai sus. Pentru fiecare coloană este calculat un bit de paritate, care este adăugat într-o nouă linie de la sfârșitul matricei. Matricea este apoi transmisă linie cu linie. La sosirea blocului, receptorul verifică toți biții de paritate. Dacă oricare din ei este greșit, va cere o retransmisie a blocului. Retransmisii succesive sunt cerute daca este nevoie, până când întregul bloc este recepționat fără erori de paritate.

Această metodă poate detecta o singură rafală de lungime n, cu numai un bit pe coloană modificat. O rafală de lungime $n+1$ va trece totuși nedetectată dacă primul și ultimul bit sunt inversați, iar toți ceilalți biți sunt corecți (o eroare în rafală nu înseamnă că toți biții sunt greșiți, ci că cel puțin primul și ultimul sunt greșiți). Dacă blocul este puternic deformat de o rafală lungă sau de rafale scurte multiple, probabilitatea ca oricare din cele n coloane să aibă, accidental, paritatea corectă este 0.5, deci probabilitatea ca un bloc eronat să fie acceptat atunci când nu ar trebui este 2^{-n}.

Cu toate că schema de mai sus poate fi uneori adecvată, în practică este larg utilizată o altă metodă: **codul polinomial** (cunoscut și sub numele de **cod cu redundanță ciclică,** eng.: **cyclic redundancy code**). Codurile polinomiale sunt bazate pe tratarea șirurilor de biți ca reprezentări de polinoame cu coeficienți 0 și 1. Un cadru de k biți este văzut ca o listă de coeficienți pentru un polinom cu k

termeni, de la x^{k-1} la x^0. Se spune că un astfel de polinom este de gradul *k*-1. Bitul cel mai semnificativ (cel mai din stânga) este coeficientul lui x^{k-1}; următorul bit este coeficientul lui x^{k-2} ş.a.m.d. De exemplu, 110001 are şase biţi şi ei reprezintă un polinom cu şase termeni cu coeficienţii 1, 1, 0, 0, 0 şi 1: $x^5+x^4+x^0$. Aritmetica polinomială este de tip modulo 2, în conformitate cu regulile teoriei algebrice. Nu există transport la adunare şi nici împrumut la scădere. Atât adunările cât şi scăderile sunt identice cu SAU EXCLUSIV. De exemplu:

10011011	00110011	11110000	01010101
+11001010	+11001101	-10100110	-10101111
01010001	11111110	01010110	11111010

Împărţirea lungă este făcută ca în binar cu excepţia faptului că scăderea este realizată modulo 2, ca mai sus. Despre un împărţitor se spune că „intră" într-un deîmpărţit dacă deîmpărţitul are tot atâţia biţi ca împărţitorul.

Atunci când este utilizată metoda codului polinomial, emiţătorul şi receptorul se pun de acord în avans asupra unui **polinom generator** *G(x)*. Atât bitul cel mai semnificativ cât şi cel mai puţin semnificativ trebuie să fie 1. Pentru a calcula **suma de control** pentru un cadru cu *m* biţi, corespunzător polinomului *M(x)*, cadrul trebuie să fie mai lung decât polinomul generator. Ideea este de a adăuga o sumă de control la sfârşitul cadrului, astfel încât polinomul reprezentat de cadrul cu sumă de control să fie divizibil prin *G(x)*. Când receptorul preia cadrul cu suma de control, încearcă să-l împartă la *G(x)*. Dacă se obţine un rest, înseamnă că a avut loc o eroare de transmisie.

Algoritmul pentru calculul sumei de control este următorul:

1. Fie *r* gradul lui *G(x)*. Se adaugă *r* biţi 0 la capătul mai puţin semnificativ al cadrului, aşa încât acesta va conţine acum *n*+*r* biţi şi va corespunde polinomului $x^rM(x)$.
2. Se împarte şirul de biţi ce corespund lui *G(x)* într-un şir de biţi corespunzând lui $x^rM(x)$, utilizând împărţirea modulo 2.
3. Se scade restul (care are întotdeauna r sau mai puţini biţi) din şirul de biţi corespunzând lui $x^rM(x)$, utilizând scăderea modulo 2. Rezultatul este cadrul cu sumă de control ce va fi transmis. Numim polinomul său *T(x)*.

Fig. 3-8 ilustrează calculul pentru cadrul 1101011011 şi $G(x) = x^4+x+1$.

Ar trebui să fie clar că *T(x)* este divizibil (modulo 2) cu *G(x)*. În orice problemă de împărţire, dacă din deîmpărţit se scade restul, atunci ceea ce rămâne este divizibil prin împărţitor. De exemplu, în baza 10, dacă împărţim 210278 la 10941 restul este 2399. Prin scăderea lui 2399 din 210278, ceea ce rămâne (207879) este divizibil cu 10941.

Să analizăm puterea acestei metode. Ce tipuri de erori vor fi detectate? Să ne imaginăm că apare o eroare de transmisie, aşa încât în loc să sosească şirul de biţi pentru *T(x)*, ajunge *T(x)* + *E(X)*. Fiecare bit din *E(x)* corespunde unui bit care a fost inversat. Dacă în *E(x)* există *k* biţi 1, aceasta înseamnă că au apărut *k* erori de un singur bit.

O singură eroare în rafală este caracterizată de un 1 iniţial, un amestec de 0 şi 1 şi un 1 final, toţi ceilalţi biţi fiind 0.

La recepţia cadrului cu sumă de control, receptorul îl împarte prin G(x); aceasta înseamnă că va calcula [*T(x)* + *E(x)*]/*G(x)*. *T(x)*/*G(x)* este 0, aşa încât rezultatul calculului este pur şi simplu *E(x)*/*G(x)*. Acele erori care se întâmplă să corespundă unor polinoame care îl au ca factor pe *G(x)* vor scăpa; toate celelalte vor fi detectate.

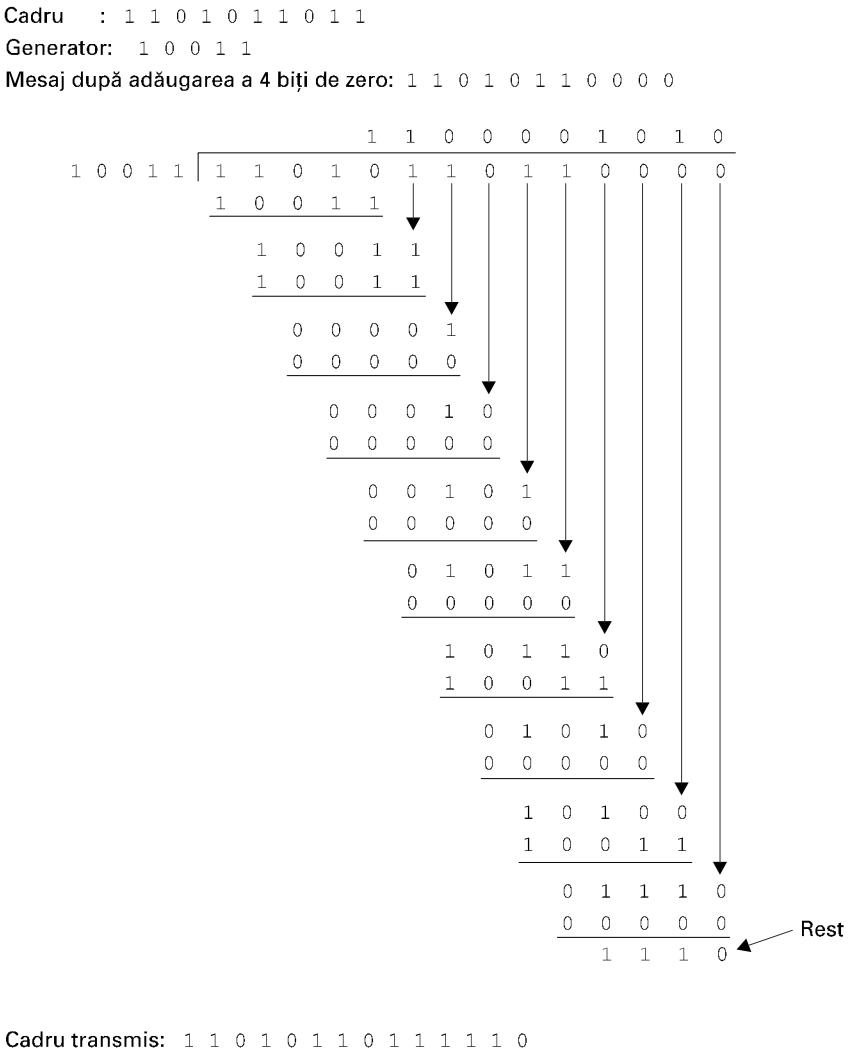


Fig. 3-8. Calculul sumei de control în cod polinomial.

Dacă a apărut o eroare pe un singur bit, $E(x) = x^i$, unde i determină care bit este eronat. Dacă $G(x)$ conține doi sau mai mulți termeni, nu poate fi divizor al lui $E(x)$, așa încât toate erorile pe un singur bit vor fi detectate.

Dacă au apărut două erori izolate pe un singur bit, atunci $E(x) = x^i + x^j$, unde i>j. Alternativ, aceasta se poate scrie ca $E(x) = x^j(x^{i-j}+1)$. Dacă presupunem că $G(x)$ nu este divizibil prin x, o condiție suficientă pentru detectarea erorilor duble este ca $G(x)$ să nu se dividă prin x^k+1 pentru orice k până la valoarea maximă $i-j$ (adică, până la lungimea maximă a cadrului). Sunt cunoscute polinoame simple, de grad mic, care asigură protecție cadrelor cu lungime mare. De exemplu, $x^{15}+x^{14}+1$ nu se va divide cu x^k+1 pentru nici o valoare a lui k mai mică decât 32768.

Dacă există un număr impar de biți eronați, $E(x)$ conține un număr impar de termeni (adică, x^5+x^2+1, dar nu x^2+1). Interesant este că, în sistemul modulo 2 nu există nici un polinom cu număr

impar de termeni care să îl aibă pe $x+1$ ca factor. Făcându-l pe $x+1$ factor al lui $G(x)$, vom putea depista toate erorile constituite dintr-un număr impar de biți inversați.

Pentru a demonstra că nici un polinom cu număr impar de termeni nu este divizibil cu $x+1$, să presupunem că $E(x)$ are un număr impar de termeni și este divizibil cu $x+1$. Factorizăm $E(x)$ în $(x+1)Q(x)$. Acum evaluăm $E(1) = (1+1)Q(1)$. Deoarece 1+1=0 (modulo 2), $E(1)$ trebuie să fie 0. Dacă $E(x)$ are un număr impar de termeni, substituind fiecare x cu 1, rezultatul obținut va fi întotdeauna 1. Prin urmare nici un polinom cu număr impar de termeni nu este divizibil cu $x+1$. În sfârșit, și cel mai important, un cod polinomial cu r biți de control va detecta toate erorile în rafală de lungime $\leq r$. O eroare în rafală de lungime k poate fi reprezentată de $x^i(x^{k-1} + \ldots + 1)$, unde i determină cât de departe este localizată rafala față de capătul din dreapta al cadrului recepționat. Dacă G(x) conține termenul x^0, atunci nu îl va avea ca factor pe x^i, așa că dacă gradul expresiei dintre paranteze este mai mic decât gradul lui $G(x)$, restul nu poate fi niciodată 0.

Dacă lungimea rafalei este $r+1$, restul împărțirii cu $G(x)$ va fi zero dacă și numai dacă rafala este identică cu $G(x)$. Prin definiția rafalei, primul și ultimul bit trebuie să fie 1, așa că potrivirea depinde de cei r - 1 biți intermediari. Dacă toate combinațiile sunt privite ca egal posibile, atunci probabilitatea ca un cadru incorect să fie acceptat ca valid este $1/2^{r-1}$.

Se poate arăta și că, dacă apare o rafală de erori mai lungă de $r+1$ biți sau dacă apar mai multe rafale mai scurte, probabilitatea ca un cadru greșit să treacă neobservat este $1/2^r$, presupunând că toate configurațiile de biți sunt la fel de probabile.

Anumite polinoame au devenit standarde internaționale. Cel folosit în IEEE 802 este:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Deși calculele necesare pentru determinarea sumei control pot să pară complicate, Peterson și Brown (1961) au arătat că pentru a calcula și verifica suma de control poate fi utilizat un simplu registru de deplasare. În practică, acest tip de circuit este utilizat aproape întotdeauna. Este folosit aproape în toate rețelele locale, și uneori și în cazul liniilor punct la punct.

Timp de zeci de ani s-a presupus că acele cadre pentru care se calculează suma de control conțin biți aleatori. Toate analizele algoritmilor pentru calculul sumei de control au fost făcute cu această presupunere. Analize mai recente ale datelor reale au arătat că această presupunere nu este corectă. Ca o consecință, în unele circumstanțe, erorile nedetectate sunt mult mai obișnuite decât s-a crezut anterior (Partridge ș.a. 1995).

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE

Pentru a face introducerea în subiectul protocoalelor, vom începe prin a analiza trei protocoale de complexitate din ce în ce mai mare. Pentru cititorii interesați, un simulator pentru aceste protocoale și pentru cele care urmează este disponibil prin Web (vezi prefața). Înainte de a analiza protocoalele, este util să explicităm unele dintre ipotezele care stau la baza modelelor de comunicație. Pentru început, considerăm că la nivelul fizic, nivelul legătură de date și nivelul rețea există procese independente care comunică transferându-și mesaje în ambele sensuri. În multe cazuri, procesele de la nivelul fizic

și de legătură de date se vor executa pe un procesor dintr-un cip special de intrare/ieșire al rețelei, iar codul nivelului rețea în CPU. Sunt posibile și alte implementări (de exemplu toate cele trei procese într-un singur cip de intrare/ieșire; nivelurile fizic și legătură de date ca proceduri apelate de procesul nivelului rețea). În orice caz, tratarea celor trei niveluri ca procese separate va face discuțiile conceptuale mai clare și de asemenea va scoate în evidență independența nivelurilor.

O altă presupunere cheie este că mașina *A* vrea să trimită un lung șir de date mașinii *B*, folosind un serviciu sigur, orientat pe conexiune. Mai târziu, vom considera cazul în care *B* vrea de asemenea să-i transmită simultan date lui *A*. Presupunem că *A* are tot timpul date gata de transmis și nu așteaptă niciodată ca aceste date să fie produse. În schimb, atunci când nivelul legătură de date al lui *A* cere date, nivelul rețea este totdeauna capabil să i le furnizeze imediat. (Și această restricție va fi abandonată mai târziu.)

Presupunem de asemenea că mașinile nu se defectează. Adică, aceste protocoale tratează erorile de transmisie, dar nu și erorile generate de defectarea sau reinițializarea calculatoarelor.

În ceea ce privește nivelul legătură de date, pachetul care trece de la nivelul rețea, prin interfață, către el este constituit din date pure, fiecare bit al acestora trebuind să fie trimis la nivelul rețea destinație. Faptul că nivelul rețea destinație poate interpreta o parte din pachetul de date ca antet nu prezintă interes pentru nivelul legătură de date.

Atunci când acceptă un pachet, nivelul legătură de date îl încapsulează într-un cadru, adăugându-i un antet și o încheiere de legătură de date (vezi fig. 3-1). Deci un cadru se compune dintr-un pachet de date, câteva informații de control (antetul), și suma de control (în încheiere) Apoi cadrul este transmis către alt nivel legătură de date. Vom presupune că există proceduri de bibliotecă adecvate pentru transmitea și recepționarea unui cadru: *to_physical_layer* și, respectiv, *from_physical_layer*. Echipamentul de transmisie calculează și adaugă suma de control (creând astfel o încheiere), astfel încât programele nivelului legătură de date nu trebuie să se preocupe de aceasta. De exemplu, ar putea fi utilizat algoritmul polinomial discutat mai devreme în acest capitol.

Inițial, receptorul nu are nimic de făcut. Doar stă așteptând să se întâmple ceva. În exemplele de protocoale din acest capitol arătăm că nivelul legătură de date așteaptă să se producă un eveniment prin apelul de procedură *wait_for_event* (*&event*). Această procedură redă controlul numai atunci când s-a întâmplat ceva (adică atunci când sosește un cadru). La revenire, variabila *event* spune ce s-a întâmplat. Mulțimea de evenimente posibile nu este aceeași pentru diferitele protocoale ce vor fi descrise și va fi definită separat, pentru fiecare protocol în parte. De reținut că, într-o situație mai realistă, nivelul legătură de date nu va sta pur și simplu în așteptarea unui eveniment, așa cum am sugerat, ci va primi o întrerupere, care îl va determina să se oprească, indiferent ce făcea în acel moment, și să se ocupe de cadrul care sosește. Cu toate acestea,, pentru simplitate, vom ignora toate detaliile activităților paralele din cadrul nivelului legătură de date și vom presupune că este tot timpul dedicat numai canalului nostru.

Când un cadru ajunge la receptor, echipamentul calculează suma de control. Dacă aceasta este incorectă (în cazul unei erori de transmisie), atunci nivelul legătură de date este informat corespunzător (*event = cksum_err*). Dacă un cadru ajunge nealterat, nivelul legătură de date este de asemenea informat (*event = frame_arrival*), așa că poate primi cadrul pentru inspecție folosind *from_physical_layer.* De îndată ce nivelul de legătură de date a primit un cadru nealterat, verifică informațiile de control din antet și dacă totul este în regulă, pachetul este transmis nivelului rețea. În nici un caz antetul nu este transmis nivelului rețea.

Există un motiv serios pentru care nivelului rețea nu trebuie să i se transmită niciodată vreo parte din antet: separarea completă a protocoalelor de rețea de cele de legătură de date. Atât timp cât

nivelul rețea nu știe nimic despre protocolul nivelului legătură de date sau despre formatul cadrului, acestea pot fi schimbate, fără să fie necesară schimbarea programelor nivelului rețea. Furnizarea unei interfețe rigide între nivelul rețea și nivelul legătură de date simplifică considerabil proiectarea programelor, deoarece protocoalele de comunicație de la niveluri diferite pot evolua independent.

```
#define MAX_PKT 1024                    /* determină dimensiunea în octeți a pachetului */

typedef enum {false, true} boolean;                                  /* tip boolean */
typedef unsigned int seq_nr;                              /* numere de secvență sau de ack */
typedef struct {unsigned char data[MAX_PKT];} packet;          /* definiția pachetului */
typedef enum {data, ack, nak} frame_kind;               /* definiția tipurilor de cadre */

typedef struct {                                 /* la acest nivel sunt transportate cadre */
 frame_kind kind;                                        /* ce fel de cadru este acesta? */
 seq_nr seq;                                                    /* număr de secvență */
 seq_nr ack;                                                  /* număr de confirmare */
 packet info;                                              /* pachetul de nivel rețea */
} frame;

/* Așteaptă producerea unui eveniment; întoarce tipul acestuia în variabila event */
void wait_for_event(event_type *event);

/* Preia un pachet de la nivelul rețea, spre a-l transmite prin canal */
void from_network_layer(packet *p);

/* Livrează nivelului rețea informația dintr-un cadru ajuns la destinație */
void to_network_layer(packet *p);

/* Preia cadrul sosit de la nivelul fizic și îl copiază în r. */
void from_physical_layer(frame *r);

/* Livrează cadrul nivelului fizic, pentru transmisie */
void to_physical_layer(frame *s);

/* Pornește ceasul și activează evenimentul timeout */
void start_timer(seq_nr k);

/* Oprește ceasul și dezactivează evenimentul timeout */
void stop_timer(seq_nr k);

/* Pornește un ceas auxiliar și activează evenimentul ack_timeout */
void start_ack_timer(void);

/* Oprește ceasul auxiliar și dezactivează evenimentul ack_timeout */
void stop_ack_timer(void);

/* Permite nivelului rețea să provoace un eveniment network_layer_ready */
void enable_network_layer(void);

/* Interzice nivelului rețea generarea unui eveniment network_layer_ready */
void disable_network_layer(void);

/* Macroinstrucțiunea inc este expandată în-line: îl incrementează circular pe k */
#define inc(k) if (k ( MAX_SEQ) k = k + 1; else k = 0
```

Fig. 3-9. Câteva definiții necesare în protocoalele care urmează. Aceste definiții se găsesc în fișierul *protocol.h*.

Fig. 3-9 arată câteva declarații (în C) comune multor protocoale ce vor fi discutate mai târziu. Sunt definite cinci structuri de date: *boolean*, *seq_nr*, *packet*, *frame_kind*, *frame*. Un *boolean* este de tip enumerativ și poate lua numai valorile *true* sau *false*. *Seq_nr* este un număr întreg mic folosit pentru a numerota cadrele, astfel încât să le putem identifica. Aceste numere de secvență sunt cuprinse între 0 și *MAX_SEQ* inclusiv, aceasta din urmă fiind o constantă definită în fiecare protocol în care este necesară. Un *pachet* este unitatea de informație schimbată între nivelul rețea și nivelul legătură de date de pe aceeași mașină sau între niveluri rețea similare. În modelul nostru el conține totdeauna *MAX_PKT* octeți, dar mai realist ar fi să aibă lungime variabilă.

Un *frame (cadru)* este compus din patru câmpuri: *kind, seq, ack*, și *info*, dintre care primele trei conțin informații de control, iar ultimul poate conține datele efective care trebuie transferate. Ansamblul acestor câmpuri de control este numit **antetul cadrului (frame header)**.

Câmpul *kind* (tip) spune dacă există sau nu date în cadru, deoarece unele protocoale fac distincție între cadrele care conțin exclusiv informații de control și cele care conțin și date. Câmpurile *seq* și *ack* sunt utilizate pentru numere de secvență și, respectiv, confirmări (*ack*nowledgements); utilizarea lor va fi descrisă în detaliu mai târziu. Câmpul *info* al unui cadru de date conține un singur pachet de date; câmpul *info* al unui cadru de control nu este utilizat. O implementare mult mai realistă va folosi un câmp *info* de lungime variabilă, omițându-l cu totul din cadrele de control.

Din nou, este important să ne dăm seama de relația dintre un pachet și un cadru. Nivelul rețea construiește un pachet luând un mesaj de la nivelul transport și adăugând la acesta antetul nivelului rețea. Acest pachet este trimis nivelului legătură de date pentru a fi inclus în câmpul *info* al unui cadru care pleacă. Când cadrul ajunge la destinație, nivelul legătură de date extrage pachetul din cadru și îl trimite nivelului rețea. În această manieră, nivelul rețea poate acționa ca și când mașinile ar putea să schimbe direct pachete.

În fig. 3-9 sunt prezentate și câteva proceduri. Acestea sunt rutine de bibliotecă ale căror detalii sunt dependente de implementare și al căror mod intern de lucru nu ne interesează în continuare. Procedura *wait_for_event* ciclează, așteptând să se întâmple ceva, așa cum am menționat mai devreme. Procedurile *to_network_layer* și *from_network_layer* sunt utilizate de nivelul de legătură de date pentru a trimite, respectiv a accepta, pachete de la nivelul de rețea. De reținut că *from_physical_layer* și *to_physical_layer* sunt utilizate pentru trimiterea cadrelor între nivelurile fizic și legătură de date. Pe de altă parte, *to_network_layer* și *from_network_layer* sunt folosite pentru a trimite pachetele între nivelul legătură de date și nivelul rețea. Cu alte cuvinte, *to_network_layer* și *from_network_layer* realizează interfața dintre nivelurile 2 și 3, în timp ce *from_physical_layer* și *to_physical_layer* realizează interfața dintre nivelurile 1 și 2.

În cele mai multe dintre protocoale se consideră că se utilizează un canal nesigur care, ocazional, poate pierde cadre întregi. Pentru a contracara efectul unor asemenea calamități, nivelul legătură de date care transmite trebuie să pornească un contor de timp sau un ceas intern de fiecare dată când trimite un cadru. Dacă nu s-a primit un răspuns într-un interval de timp predefinit, la expirarea acestuia, nivelul legătură de date primește un semnal de întrerupere.

În protocoalele noastre acest lucru este asigurat de procedura *wait_for_event* care întoarce *event = timeout*. Procedurile *start_timer* și *stop_ timer* sunt utilizate pentru a porni, respectiv a opri contorul de timp. Expirarea timpului este posibilă numai atunci când contorul de timp este pornit. Este permis explicit să se apeleze *start_ timer* în timp ce contorul de timp lucrează; un astfel de apel va reseta pur și simplu contorul de timp, determinându-l să genereze următorul semnal de expirare de timp după ce întregul interval de timp se va epuiza (exceptând cazul în care este resetat sau dezactivat în acest interval timp).

Procedurile *start_ack_timer* și *stop_ack_timer* sunt folosite pentru a controla un contor de timp auxiliar, utilizat pentru a genera confirmări în anumite condiții. Procedurile *enable_network_layer* și *disable_network_layer* sunt utilizate în protocoalele mai sofisticate, în care nu mai presupunem că nivelul de rețea are tot timpul pachete de trimis. Când nivelul legătură de date activează nivelul rețea, acestuia i se permite să întrerupă atunci când are un pachet de trimis. Indicăm aceasta cu *event* = *network_layer_ready*. Când un nivel rețea este dezactivat, acesta nu poate provoca asemenea evenimente. Gestionând cu prudență activarea și dezactivarea nivelului rețea, nivelul legătură de date îl poate împiedica pe acesta să-l inunde cu pachete atunci când nu mai are spațiu în tampon.

Numerele de secvență ale cadrelor sunt întotdeauna de la 0 la *MAX_SEQ* (inclusiv), unde *MAX_SE*Q diferă de la protocol la protocol. Deseori este necesar ca numărul de secvență să avanseze circular (adică *MAX_SEQ* este urmat de 0). Această incrementare este realizată de macroinstrucțiunea *inc*. A fost definită ca macroinstrucțiune, deoarece este utilizată in-line în secvența critică. Așa cum vom vedea mai târziu, factorul care limitează performanța rețelei este adesea prelucrarea efectuată de protocol, așa că definirea operațiilor simple, ca aceasta, ca macroinstrucțiuni, nu afectează claritatea codului, dar îmbunătățește performanța. Mai mult, de vreme ce *MAX_SEQ* va avea diferite valori în diferite protocoale, definirea de macroinstrucțiuni face posibilă includerea tuturor protocoalelor în același fișier binar fără conflict. Această posibilitate este utilă pentru simulator.

Declarațiile din fig. 3-9 fac parte din fiecare dintre protocoalele care urmează. Pentru a economisi spațiu și pentru a furniza referințe convenabile, acestea au fost extrase și listate împreună, dar din punct de vedere conceptual ele trebuie incluse în protocoalele respective. În C, aceasta se realizează punând definițiile într-un fișier antet special, în acest caz *protocol.h,* și utilizând facilitatea #include a preprocesorului C pentru a le include în fișierele protocol.

3.3.1 Un protocol simplex fără restricții

Ca un prim exemplu vom considera cel mai simplu protocol posibil. Datele sunt transmise într-o singură direcție. Cele două niveluri rețea, de transmisie și de recepție, sunt considerate tot timpul pregătite. Timpul de prelucrare poate fi ignorat. Memoria de stocare disponibilă este infinită. Și, cel mai bun lucru dintre toate, canalul de comunicație între niveluri legătură de date nu pierde și nu alterează niciodată cadrele. Acest protocol total nerealist, pe care îl vom numi "utopia", este prezentat în fig. 3-10. Protocolul constă din două proceduri distincte, una de emisie și cealaltă de recepție. Emițătorul lucrează la nivelul legătură de date al mașinii sursă, iar receptorul la nivelul legătură de date al mașinii de destinație. Nu se folosesc nici numere de secvență, nici confirmări, așa că nu este nevoie de *MAX_SEQ*. Singurul eveniment posibil este *frame_arrival* (sosirea unui cadru nealterat).

Emițătorul este într-un ciclu infinit care doar inserează datele pe linie cât poate de repede. Ciclul constă din trei acțiuni: preluarea unui pachet de date de la nivelul rețea (care este întotdeauna serviabil), construirea unui cadru de ieșire folosind variabila *s* și trimiterea cadrului pe drumul său. Acest protocol utilizează numai câmpul *info* al cadrului, deoarece celelalte câmpuri se referă la erori și secvențe de control, iar în cazul nostru nu există erori sau restricții de control.

Receptorul este la fel de simplu. Inițial el așteaptă să se întâmple ceva, singura posibilitate fiind sosirea unui cadru nealterat. În cele din urmă, cadrul ajunge, iar procedura *wait_for_event* se întoarce cu *event* setat la *frame_arrival* (care este oricum ignorat). Apelul rutinei *from_physical_layer* mută cadrul nou sosit din zona tampon a echipamentului în variabila *r*. În cele din urmă pachetul de date este trimis nivelului rețea și nivelul legătură de date revine la starea de așteptare a cadrului următor, autosuspendându-se pur și simplu până la sosirea unui nou cadru.

```
/* Protocolul 1 (utopia) asigură transmitere de date doar într-o direcție, de la transmițător la receptor. Canalul de comunicație se presupune a fi fără erori, iar despre receptor se presupune că este capabil să prelucreze infinit de repede tot ce primește de la intrare. Deci, transmițătorul nu face decât să stea într-o buclă, pompând date pe linie cât de repede poate */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
 frame s;                                        /* tampon pentru cadrul transmis */
 packet buffer;                                  /* tampon pentru pachetul transmis */

 while (true) {
   from_network_layer(&buffer);                  /* preia ceva de transmis */
   s.info = buffer;                              /* îl copiază în s pentru transmitere */
   to_physical_layer(&s);                        /* îl trimite pe traseu */
 }
/* Tomorrow, and tomorrow, and tomorrow

Creeps in this petty pace from day to day

To the last syllable of recorded time

- Macbeth, V, v */
 }

 void receiver1(void)
 {
  frame r;
  event_type event;                      /* completat de rutina wait, dar neutilizat aici */

  while (true) {
    wait_for_event(&event);              /* singura posibilitate este sosirea unui cadru */
    from_physical_layer(&r);                          /* preia cadrul sosit */
    to_network_layer(&r.info);                        /* predă datele nivelului rețea */
  }
 }
```

Fig. 3-10. Un protocol simplex fără restricții.

3.3.2 Un protocol simplu Stop-and-Wait (pas-cu-pas)

Acum vom renunța la cea mai nerealistă restricție utilizată în protocolul 1: posibilitatea ca nivelul rețea receptor să prelucreze datele de intrare cu viteză infinită (sau echivalent, prezența în nivelul legăturii de date receptor a unui tampon infinit în care să fie memorate, cât timp își așteaptă rândul, toate cadrele sosite). Totuși, se presupune în continuare că nu se produc erori pe canalul de comunicație și că traficul de date este încă simplex.

Principala problemă pe care trebuie să o rezolvăm aici este cum să evităm ca emițătorul să inunde receptorul cu date care sosesc mai rapid decât poate acesta să prelucreze. În esență, dacă receptorul are nevoie de un timp Δt ca să execute *from_physical_layer* și *to_network_layer*, atunci emițătorul trebuie să transmită la o viteză medie mai mică de un cadru la fiecare interval de timp de Δt. Mai mult, dacă presupunem că echipamentul receptor nu realizează automat memorarea în zona tam-

pon și gestiunea cozii de așteptare, atunci emițătorul nu trebuie să transmită niciodată un nou cadru până când cel vechi nu a fost preluat de rutina *from_physical_layer*, ca nu cumva cel nou să se scrie peste cel vechi.

În anumite situații speciale (de exemplu, transmisie sincronă și un nivel legătură de date receptor complet dedicat prelucrării unei singure linii de intrare) ar putea fi posibil ca emițătorul să introducă pur și simplu o întârziere în protocolul 1, pentru a-l încetini suficient, astfel încât să se evite inundarea receptorului. Totuși, de obicei, fiecare nivel legătură de date va avea mai multe linii de luat în considerare și intervalul de timp între sosirea unui cadru și începutul prelucrării sale poate varia considerabil. Dacă cei ce proiectează rețele pot calcula comportamentul receptorului în cazul cel mai defavorabil, atunci pot programa emițătorul să transmită atât de încet, încât, chiar dacă fiecare cadru va suferi întârzierea maximă, nu vor exista depășiri. Problema cu această abordare este aceea că este prea conservatoare. Ea conduce la o utilizare a lărgimii de bandă care este cu mult sub optim, cu excepția situației în care cazurile cel mai favorabil și cel mai defavorabil sunt aproape la fel (adică, variația timpului de reacție al nivelului legătură de date este mică).

```
/* Protocolul 2 (stop-and-wait) asigură la rândul său un flux de date unidirecțional, de la
emițător la receptor. Despre canalul de comunicație se presupune din nou că este fără
erori, ca și în protocolul 1. Totuși, de data aceasta, receptorul are doar un tampon de ca-
pacitate limitată și o viteză de prelucrare finită, așa că protocolul trebuie să împiedice
în mod explicit emițătorul să inunde receptorul cu date mai repede decât le poate trata
acesta. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
 frame s;                                              /* tampon pentru cadrul trimis */
 packet buffer;                                       /* tampon pentru pachetul trimis */
 event_type event;                        /* singura posibilitate este sosirea unui cadru */

 while (true) {
   from_network_layer(&buffer);                              /* preia ceva de transmis */
   s.info = buffer;                                 /* îl copiază în s pentru transmitere */
   to_physical_layer(&s);                                    /* la revedere, micuț cadru */
   wait_for_event(&event);                        /* nu continuă până nu primește semnalul */
 }
}

void receiver2(void)
{
 frame r, s;                                           /* zone tampon pentru cadre */
 event_type event;                        /* singura posibilitate este sosirea unui cadru */

 while (true) {
   wait_for_event(&event);                /* singura posibilitate este sosirea unui cadru */
   from_physical_layer(&r);                                  /* preia cadrul sosit */
   to_network_layer(&r.info);                          /* livrează datele nivelului rețea */
   to_physical_layer(&s);           /* trimite un cadru fictiv pentru a trezi emițătorul */
 }
}
```

Fig. 3-11. Un protocol simplex stop-and-wait.

O soluție mult mai generală a acestei dileme este ca receptorul să furnizeze o reacție către emițător. După trimiterea unui pachet către nivelul său rețea, receptorul trimite un mic cadru fictiv către emițător care, de fapt, îi dă emițătorului permisiunea să transmită următorul cadru. După ce a transmis un cadru, emițătorul este obligat de protocol să intre în așteptare un timp, până când sosește micul cadru fictiv (deci confirmarea). Utilizarea reacției de la receptor pentru a anunța emițătorul că poate trimite date este un exemplu de control al fluxului menționat anterior.

Protocoalele în care emițătorul trimite un cadru și apoi, înainte de a continua, așteaptă o confirmare, se numesc **stop-and-wait** (pas-cu-pas). Fig. 3-11 prezintă un exemplu de protocol simplex *stop-and-wait*. Chiar dacă traficul de date este simplex, mergând numai de la emițător la receptor, cadrele se deplasează în ambele direcții. În consecință, canalul de comunicație dintre cele două niveluri legătură de date trebuie să permită transferul de informație bidirecțional. Totuși, acest protocol impune o alternanță strictă a fluxului: mai întâi emițătorul trimite un cadru, apoi receptorul trimite un alt cadru, apoi emițătorul trimite alt cadru ș.a.m.d. În acest caz este suficient un canal fizic semiduplex.

Ca și în protocolul 1, emițătorul începe prin preluarea unui pachet de la nivelul rețea, utilizarea lui pentru construirea unui cadru și trimiterea acestuia. Numai că acum, spre deosebire de protocolul 1, emițătorul trebuie să aștepte până când sosește un cadru de confirmare, înainte de a relua ciclul și a prelua următorul pachet de la nivelul rețea. Nivelul legătură de date care transmite nu are nevoie să inspecteze cadrul care sosește: nu există decât o singură posibilitate. Cadrul primit este întotdeauna o confirmare.

Singura diferență dintre *receiver1* și *receiver2* este aceea că după transmiterea unui pachet către nivelul rețea, *receiver2* trimite un cadru de confirmare înapoi la emițător, înainte de a intra din nou în bucla de așteptare. Deoarece numai sosirea cadrului de întoarcere la emițător este importantă, nu și conținutul lui, receptorul nu trebuie să pună nici o informație particulară în el.

3.3.3 Un protocol simplex pentru un canal cu zgomote

Să considerăm situația normală a unui canal de comunicație care pot apărea erori. Cadrele pot fi fie modificate, fie complet pierdute. Totuși, presupunem că dacă un cadru a fost modificat în tranzit, echipamentul receptor va detecta acest lucru atunci când calculează suma de control. Dacă un cadru este modificat într-un asemenea mod, încât suma de control este totuși corectă, situație care este foarte puțin probabilă, acest protocol (și toate celelalte protocoale) pot eșua (adică, trimit un pachet incorect către nivelul rețea).

La prima vedere s-ar părea că o variantă a protocolului 2 va funcționa: adăugarea unui contor de timp (ceas). Emițătorul poate trimite un cadru, dar receptorul va trimite un cadru de confirmare numai dacă informația a fost recepționată corect. Dacă la receptor ajunge un cadru modificat, el va fi eliminat. După un timp, emițătorul va ieși din așteptare și va retrimite cadrul. Acest proces va fi repetat până când cadrul va ajunge în final intact. Schema de mai sus conține o eroare fatală. Gândiți-vă la problemă și încercați să descoperiți ce este greșit înainte să citiți mai departe.

Pentru a vedea ce poate merge rău, amintiți-vă care este sarcina proceselor nivelului legătură de date - aceea de a asigura comunicație fără erori, transparentă, între procesele nivelului rețea. Nivelul rețea de pe mașina *A* dă o serie de pachete nivelului său legătură de date, care trebuie să asigure o serie identică de pachete nivelului rețea de pe mașina *B* prin nivelul său legătură de date. În particular, nivelul rețea de pe *B* nu are nici o posibilitate să știe că un pachet a fost pierdut sau duplicat, așa că nivelul legătură de date trebuie să garanteze că nici o combinație de erori de transmisie, indiferent cât de puțin probabile, nu poate produce un pachet duplicat care să fie transmis nivelului rețea.

Să considerăm următorul scenariu:

1. Nivelul rețea de pe *A* trimite pachetul 1 către nivelul său legătură de date. Pachetul este corect recepționat de *B* și este trimis nivelului rețea de pe *B*. *B* trimite un cadru de confirmare înapoi lui *A*.
2. Cadrul de confirmare s-a pierdut complet. El nu va mai ajunge deloc. Viața ar fi cu mult mai simplă în cazul în care canalul ar altera sau pierde doar cadre de date, nu și cadre de control, dar, din nefericire, canalul nu face discriminări.
3. Nivelul de legătură de date de pe *A* așteaptă expirarea timpului limită. Nerecepționând o confirmare, el presupune (incorect) că acel cadru de date a fost modificat sau pierdut și trimite încă o dată cadrul conținând pachetul 1.
4. Cadrul duplicat ajunge și el cu bine la nivelul legătură de date *B* și este trimis nivelului rețea de acolo. Dacă *A* trimite un fișier lui *B*, o porțiune de fișier va fi duplicată (adică, copia fișierului făcută de *B* va fi incorectă și eroarea nu va fi detectată). Cu alte cuvinte, protocolul va eșua.

În mod clar, este necesară o soluție ca receptorul să poată distinge un cadru pe care îl vede pentru prima dată de o retransmisie. Soluția evidentă este aceea ca emițătorul să pună un număr de secvență în antetul fiecărui cadru pe care îl trimite. Apoi receptorul poate verifica numărul de secvență al fiecărui cadru sosit pentru a vedea dacă este un cadru nou sau un duplicat ce trebuie eliminat.

Deoarece este de dorit ca un antet de cadru să fie de dimensiune mică, se pune întrebarea următoare: care este numărul minim de biți necesari pentru numărul de secvență? Singura ambiguitate în acest protocol este între un cadru m și succesorul său $m+1$. În cazul în care cadrul m este pierdut sau modificat, receptorul nu îl va confirma, așa încât emițătorul va încerca să-l retransmită. Odată ce a fost corect recepționat, receptorul va trimite o confirmare înapoi la emițător. Aici este punctul în care putem să avem necazuri. După cum cadrul de confirmare ajunge sau nu corect înapoi la emițător, emițătorul va încerca să transmită m sau $m+1$.

Evenimentul care determină emițătorul să înceapă transmiterea lui $m+2$ este sosirea unei confirmări pentru $m+1$. Dar aceasta presupune că m a fost recepționat corect și, mai mult, confirmarea a fost de asemenea corect recepționată de emițător (altfel emițătorul nu ar fi trimis $m+1$, ca să nu mai vorbim de $m+2$). În consecință, singura ambiguitate este între un cadru și predecesorul sau succesorul său imediat, nu între ultimii doi.

Este deci suficient un număr de secvență de 1 bit (0 sau 1). La fiecare moment de timp, receptorul așteaptă un anumit număr de secvență. Orice cadru sosit, care conține un număr de secvență greșit este rejectat ca duplicat. Atunci când sosește un cadru cu număr de secvență corect, acesta este acceptat și transmis nivelului rețea. Apoi numărul de secvență așteptat este incrementat modulo 2 (adică, 0 devine 1 și 1 devine 0).

Un exemplu de astfel de protocol este prezentat în fig. 3-12. Protocoalele în care emițătorul așteaptă pentru o confirmare pozitivă înaintea de a trece la următorul element de date se numesc deseori **PAR (Positive Acknowledgement with Retransmision**, rom.: confirmare pozitivă cu retransmitere) sau **ARQ (Automatic Repeat reQuest**, rom: cerere automată de repetare). Asemenea protocolului 2, și acesta transmite datele într-o singură direcție.

Protocolul 3 se deosebește de predecesorii săi prin aceea că și emițătorul și receptorul au o variabilă a cărei valoare este păstrată cât timp nivelul legătură de date este în starea de așteptare. Emițătorul păstrează numărul de secvență al următorului cadru de transmis în *next_frame_to_send*; receptorul păstrează numărul de secvență al următorului cadru așteptat în *frame_expected*. Fiecare protocol are o scurtă fază de inițializare înainte de a intra în bucla infinită.

```
/* Protocolul 3 (par) permite un flux de date unidirecțional, printr-un canal nesigur. */

#define MAX_SEQ 1                                   /* trebuie să fie 1 pentru protocolul 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
 seq_nr next_frame_to_send;          /* numărul de secvență al următorului cadru trimis */
 frame s;                                                    /* variabilă temporară */
 packet buffer;                                    /* tampon pentru pachetul transmis */
 event_type event;

 next_frame_to_send = 0;                /* inițializează numerele de secvență de ieșire */
 from_network_layer(&buffer);                                /* preia primul pachet */
 while (true) {
   s.info = buffer;                           /* construiește un cadru pentru transmitere */
   s.seq = next_frame_to_send; /* inserează în cadru un număr de secvență */
   to_physical_layer(&s);                                   /* îl trimite pe traseu */
   start_timer(s.seq);           /* dacă răspunsul întârzie prea mult, timpul va expira */
   wait_for_event(&event);                        /* frame_arrival, cksum_err, timeout */
   if (event == frame_arrival) {                              /* preia confirmarea */
     from_physical_layer(&s);
    if(s.ack==next_frame_to_send) {
      stop_timer(s.ack);                                        /* oprește ceasul */
      from_network_layer(&buffer);               /* preia următorul pachet de transmis */
      inc(next_frame_to_send);                     /* inversează next_frame_to_send */
   }
 }
 }
}

void receiver3(void)
{
 seq_nr frame_expected;
 frame r, s;
 event_type event;

 frame_expected = 0;
 while (true) {
   wait_for_event(&event);                       /* variante: frame_arrival, cksum_err */
   if (event == frame_arrival) {                            /* a sosit un cadru corect */
     from_physical_layer((r);                                /* preia cadrul nou sosit */
     if (r.seq == frame_expected) {                 /* este cel pe care îl așteptam */
       to_network_layer((r.info);                 /* transferă datele nivelului rețea */
       inc(frame_expected);                        /* incrementează număr de secvență */
     }
     s.ack=1-frame_expected;                                 /* spune care frame este confirmat */
     to_physical_layer(&s);               /* nici unul dintre câmpuri nu este utilizat */
   }
 }
}
```

Fig. 3-12. Un protocol cu confirmare pozitivă și retransmitere.

După transmiterea unui cadru, emițătorul declanșează contorul de timp. Dacă acesta era deja pornit, atunci va fi resetat pentru un nou interval complet. Intervalul de timp trebuie să fie ales astfel, încât să permită sosirea cadrului la receptor, prelucrarea sa de către receptor, chiar și în cazul cel mai defavorabil, și propagarea cadrului de confirmare înapoi la emițător. Numai atunci când a expirat acest interval de timp se poate spune cu siguranță că s-a pierdut fie cadrul transmis, fie confirmarea sa și se poate trimite un duplicat. Dacă intervalul de timp este prea mic, emițătorul va transmite cadre care nu sunt necesare. Aceste cadre nu influențează corectitudinea protocolului, dar îi afectează performanțele.

După transmiterea unui cadru și pornirea contorului de timp, emițătorul așteaptă să se întâmple ceva interesant. Există doar trei posibilități: un cadru de confirmare ajunge intact, sosește un cadru de confirmare eronat sau expiră timpul. Dacă sosește o confirmare validă, atunci emițătorul preia următorul pachet de la nivelul rețea și îl pune în tampon, scriind peste pachetul anterior. De asemenea avansează numărul de secvență. Dacă sosește un cadru modificat sau nu sosește nici un cadru, nu se modifică nici tamponul și nici numărul de secvență, așa că poate fi transmis un duplicat.

Atunci când la receptor sosește un cadru corect, este verificat numărul de secvență, pentru a vedea dacă nu cumva este un duplicat. Dacă nu, este acceptat, transmis nivelului rețea și este generată o confirmare. Cadrele duplicate și cele modificate nu sunt trimise către nivelul rețea.

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ

În protocoalele anterioare, cadrele cu date erau transmise într-o singură direcție. În cele mai multe situații practice, este necesar să se transmită date în ambele direcții. O modalitate de a realiza transmisia de date full-duplex este de a avea două canale de comunicație separate, fiecare dintre ele fiind utilizat pentru traficul de date simplex (în direcții diferite). Dacă se face aceasta, vom avea două circuite fizice separate, fiecare cu un canal "direct" (eng.: forward) pentru date și un canal "invers" (eng.: reverse) pentru confirmări. În ambele cazuri lărgimea de bandă a canalului invers este irosită aproape în totalitate. Ca efect, utilizatorul plătește pentru două circuite, dar utilizează doar capacitatea unuia.

O idee mai bună este să se utilizeze același circuit pentru date în ambele direcții. În definitiv, la protocoalele 2 și 3 a fost deja utilizată transmiterea cadrelor în ambele sensuri și canalul invers are aceeași capacitate ca și canalul direct. În acest model, cadrele cu date de la *A* la *B* sunt amestecate cu cadrele de confirmare de la *A* la *B*. Uitându-se la câmpul *kind* din antetul cadrului ce a sosit, receptorul poate spune dacă este vorba de un cadru de date sau de confirmare.

Cu toate că întrepătrunderea cadrelor de date și control pe același circuit constituie o îmbunătățire față de cazul utilizării a două circuite fizice separate, mai este posibilă încă o îmbunătățire. Atunci când sosește un cadru cu date, în locul emiterii imediate a unui cadru de control separat, receptorul stă și așteaptă până când nivelul rețea îi dă următorul pachet. Confirmarea este atașată cadrului cu date de ieșire (utilizând câmpul *ack* din antetul cadrului). De fapt, confirmarea este transportată pe gratis de către următorul cadru cu date de ieșire. Tehnica întârzierii confirmării, astfel încât să poată fi agățată de următorul cadru de date, este cunoscută ca **atașare** (eng.: **piggybacking**).

Principalul avantaj al utilizării tehnicii de atașare în comparație cu utilizarea cadrelor de confirmare distincte este o mai bună utilizare a lărgimii de bandă disponibile. Câmpul *ack* din antetul ca-

drului ocupă doar câțiva biți, în timp ce un cadru separat va necesita un antet, confirmarea și o sumă de control. În plus, mai puține cadre transmise înseamnă mai puține întreruperi datorate "sosirii cadrelor" și probabil mai puține zone tampon în receptor, în funcție de modul în care sunt organizate programele receptorului. În următorul protocol ce va fi examinat, câmpul de atașare ocupă doar un bit în antetul cadrului. Arareori ocupă mai mult de câțiva biți.

Totuși, tehnica de atașare introduce o complicație care nu era prezentă în cazul confirmărilor separate. Cât timp trebuie să aștepte nivelul legătură de date pachetul la care să atașeze confirmarea? Dacă nivelul legătură de date așteaptă mai mult timp decât perioada de timeout a emițătorului, cadrul va fi retransmis, anulând complet rolul confirmărilor. Dacă nivelul legătură de date ar fi un oracol și ar putea prezice viitorul, ar putea ști când va sosi următorul pachet de la nivelul rețea și ar putea decide dacă să îl aștepte sau să trimită imediat o confirmare separată, în funcție de cât de lungă urmează să fie așteptarea. Desigur, nivelul legătură de date nu poate prezice viitorul, așa că trebuie să recurgem la câteva scheme ad-hoc, cum ar fi așteptarea pentru un număr fixat de milisecunde. Dacă un nou pachet sosește repede, confirmarea este adăugată în el; altfel, dacă până la sfârșitul acestei perioade de timp nu a sosit un nou pachet, nivelul legătură de date trimite un cadru de confirmare separat.

Următoarele trei protocoale sunt protocoale bidirecționale care aparțin unei clase de protocoale numite protocoale cu **fereastră glisantă** (eng.: **sliding window**). Cele trei diferă între ele în termeni de eficiență, complexitate și necesar de tampoane, așa cum vom vedea mai târziu. În cadrul acestora, ca în toate protocoalele cu fereastră glisantă, fiecare cadru expediat conține un număr de secvență cuprins între 0 și o valoare maximă. Maximul este de obicei 2^n -1, ca numărul de secvență să se încadreze exact într-un câmp de n biți. Protocoalele cu fereastră glisantă pas-cu-pas utilizează $n=1$, restricționând numerele de secvență la 0 și 1, dar versiuni mai sofisticate pot utiliza o valoare arbitrară a lui n.

Esența protocoalelor cu fereastră glisantă este aceea că, la orice moment de timp, emițătorul menține o mulțime de numere de secvență care corespund cadrelor pe care are permisiunea să le trimită. Se spune că aceste cadre aparțin **ferestrei de transmisie** (eng.: **sending window**). Similar, receptorul menține de asemenea o **fereastră de recepție** (eng.: **receiving window**), ce corespunde mulțimii de cadre care pot fi acceptate. Fereastra emițătorului și fereastra receptorului nu trebuie să aibă aceleași limite minime și maxime și nici măcar aceeași dimensiune. În unele protocoale ele au dimensiune fixă, dar în altele ele pot crește sau scădea pe măsură ce cadrele sunt emise sau recepționate.

Chiar dacă aceste protocoale dau nivelului legătură de date mai multă independență în ceea ce privește ordinea în care poate primi sau recepționa cadre, nu am renunțat la cerința ca protocolul să livreze pachetele la nivelul rețea destinație în aceeași ordine în care acestea sunt trimise către nivelul legătură de date de pe mașina emițătoare. Nu am modificat nici condiția impusă canalului fizic de comunicație, care trebuie să se comporte "ca un fir", adică trebuie să trimită toate cadrele în ordinea emiterii.

Numerele de secvență din cadrul ferestrei emițătorului reprezintă cadre transmise sau cadre ce pot fi transmise, dar încă neconfirmate. De fiecare dată când de la nivelul rețea sosește un nou pachet, acestuia îi este atribuit următorul număr de secvență, iar marginea superioară a ferestrei este avansată cu unu. Atunci când sosește o confirmare, crește cu unu limita inferioară a ferestrei. În acest mod, fereastra menține continuu o listă de cadre neconfirmate. Un exemplu este prezentat în fig. 3-13.

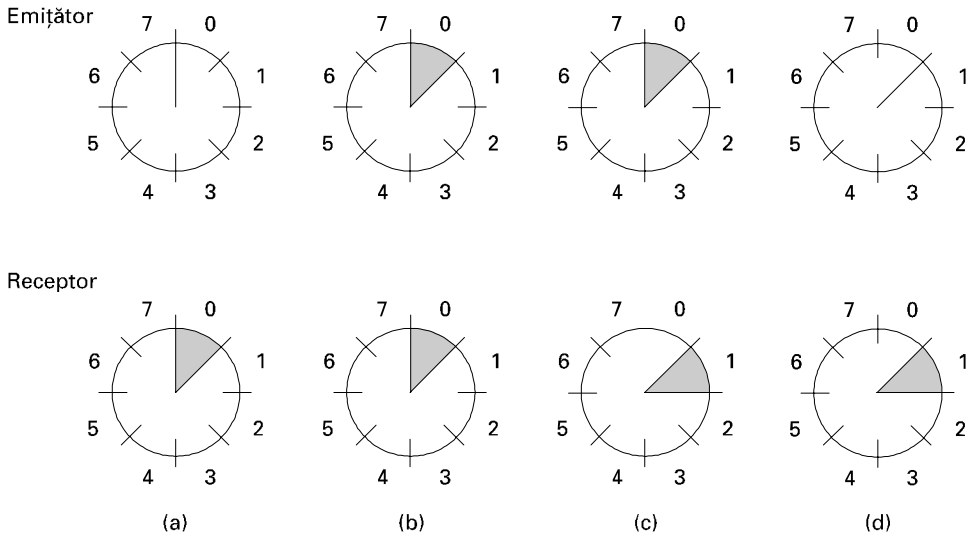


Fig. 3-13. O fereastră glisantă de dimensiune 1, cu număr de secvență de 3 biți. (a) Inițial. (b) După ce a fost transmis primul cadru. (c) După ce a fost recepționat primul cadru. (d) După ce a fost recepționată prima confirmare.

Deoarece cadrele din fereastra curentă a emițătorului pot fi pierdute sau modificate pe traseu, emițătorul trebuie să păstreze toate aceste cadre în memoria sa pentru o posibilă retransmisie. Astfel, dacă dimensiunea maximă a ferestrei este *n*, emițătorul are nevoie de *n* tampoane pentru a păstra cadrele neconfirmate. Dacă fereastra crește la dimensiunea maximă, nivelul legătură de date al emițătorului trebuie să forțeze închiderea nivelului rețea până când se eliberează un tampon.

Fereastra nivelului legătură de date receptor corespunde cadrelor pe care acesta le poate accepta. Orice cadru din afara ferestrei este eliminat fără comentarii. Atunci când este recepționat un cadru al cărui număr de secvență este egal cu marginea inferioară a ferestrei, acesta este trimis nivelului rețea, este generată o confirmare și fereastra se deplasează cu o unitate. Spre deosebire de fereastra emițătorului, fereastra receptorului rămâne întotdeauna la dimensiunea inițială. De notat că o fereastră de dimensiune 1 înseamnă că nivelul legătură de date acceptă numai cadre ordonate, dar pentru ferestre mari afirmația nu mai este valabilă. Nivelul rețea este, dimpotrivă, alimentat întotdeauna cu date în ordine corectă, indiferent de dimensiunea ferestrei nivelului legătură de date.

Fig. 3-13 prezintă un exemplu cu o fereastră de dimensiune maximă 1. Inițial, nu sunt emise cadre, așa că marginile inferioară și superioară ale ferestrei emițătorului sunt egale, dar o dată cu trecerea timpului, situația evoluează ca în figură.

3.4.1 Un protocol cu fereastră glisantă de un bit

Înainte de a analiza cazul general, să examinăm mai întâi un protocol cu fereastră glisantă având dimensiunea maximă a ferestrei 1. Un astfel de protocol utilizează metoda stop-and-wait, deoarece emițătorul transmite un cadru și așteaptă confirmarea sa înaintea transmiterii următorului cadru.

Fig. 3-14 prezintă un astfel de protocol. Ca și alte protocoale, acesta începe prin definirea unor variabile. *Next_frame_to_send* arată ce cadru încearcă să transmită emițătorul. Similar, *frame_ expected* arată ce cadru este așteptat de receptor. În ambele cazuri singurele posibilități sunt 0 și 1.

```
/* Protocolul 4 (fereastră glisantă) este bidirecțional. */

#define MAX_SEQ 1                                   /* pentru protocolul 4 trebuie să fie 1 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4(void)
{
 seq_nr next_frame_to_send;                                          /* doar 0 sau 1 */
 seq_nr frame_expected;                                              /* doar 0 sau 1 */
 frame r, s;                                                  /* variabile temporare */
 packet buffer;                                 /* pachetul curent, care este transmis */
 event_type event;

 next_frame_to_send = 0;                          /* următorul cadru pe fluxul de ieșire */
 frame_expected = 0;                          /* numărul de secvență al cadrului așteptat */
 from_network_layer(&buffer);                     /* preia un pachet de la nivelul rețea */
 s.info = buffer;                               /* pregătește trimiterea cadrului inițial */
 s.seq = next_frame_to_send;                    /* inserează în cadru numărul de secvență */
 s.ack = 1 - frame_expected;                                  /* confirmare atașată */
 to_physical_layer(&s);                                         /* transmite cadrul */
 start_timer(s.seq);                                             /* pornește ceasul */
 while (true) {
   wait_for_event(&event);                      /* frame_arrival, cksum_err sau timeout */
   if (event == frame_arrival) {                          /* un cadru a ajuns nedeformat */
     from_physical_layer(&r);                                        /* preia cadrul */
     if (r.seq == frame_expected) {                  /* tratează fluxul de cadre sosite */
       to_network_layer(&r.info);                     /* predă pachetul nivelului rețea */
       inc(frame_expected);                  /* inversează numărul de secvență așteptat */
     }
     if (r.ack == next_frame_to_send)            { /* tratează fluxul de cadre de ieșire */
       stop_timer(r.ack);                                          /* oprește ceasul */
       from_network_layer(&buffer);           /* preia un nou pachet de la nivelul rețea */
       inc(next_frame_to_send);         /*inversează numărul de secvență al emițătorului */
     }
   }
   s.info = buffer;                                    /* construiește cadrul care iese */
   s.seq = next_frame_to_send;                         /* inserează un număr de secvență */
   s.ack = 1 - frame_expected;          /* numărul de secvență al ultimului cadru primit */
   to_physical_layer(&s);                                       /* transmite un cadru */
   start_timer(s.seq);                                             /* pornește ceasul */
 }
}
```

Fig. 3-14. Un protocol cu fereastră glisantă de 1 bit.

În mod normal, unul dintre cele două niveluri legătură de date pornește primul trimițând primul cadru. Cu alte cuvinte, numai unul din programele nivelului legătură de date va conține apelurile procedurilor *to_physical_layer* și *start_timer* în afara buclei principale. În eventualitatea că ambele niveluri legătură de date pornesc simultan, apare o situație specială, care va fi discutată mai târziu. Mașina care pornește prima preia primul pachet de la nivelul rețea propriu, construiește din el un cadru și îl trimite. Când acest cadru (sau oricare altul) sosește, nivelul legătură de date receptor verifică dacă nu cumva este un duplicat, exact ca în protocolul 3. Dacă respectivul cadru este cel așteptat, atunci este trimis nivelului rețea și fereastra receptorului este deplasată.

Câmpul de confirmare conține numărul ultimului cadru recepționat fără eroare. Dacă acest număr corespunde cu numărul de secvență al cadrului pe care emițătorul încearcă să-l transmită, emițătorul știe că a terminat cu cadrul memorat în tampon și poate prelua următorul pachet de la nivelul său rețea. Dacă numărul de secvență nu corespunde, el trebuie să continue să trimită același cadru. De fiecare dată când este recepționat un cadru, un alt cadru este trimis de asemenea înapoi.

Să examinăm acum cât de fiabil este protocolul 4 la condițiile limită. Să presupunem că *A* încearcă să trimită cadrul 0 lui *B* și *B* încearcă să trimită cadrul său 0 lui *A*. Să presupunem că *A* trimite un cadru lui *B*, dar timpul de expirare al lui *A* este puțin prea scurt. În consecință, datorită expirării repetate a timpului, *A* va trimite o serie de cadre identice, toate cu *seq*=0 și *ack*=1.

Atunci când la *B* sosește primul cadru corect, el va fi acceptat și *frame_expected* va fi setat la 1. Toate cadrele următoare vor fi respinse, deoarece B așteaptă cadre cu numărul de secvență 1, nu 0. Mai mult, deoarece toate duplicatele au *ack*=1 și B este încă în așteptarea confirmării lui 0, B nu va prelua un nou pachet de la nivelul său rețea.

După sosirea fiecărui duplicat respins, B trimite lui A un cadru conținând *seq*=0 și *ack*=0. În cele din urmă, unul dintre acestea sosește corect la *A,* făcându-l pe *A* să înceapă să trimită următorul pachet. Nici o combinație de cadre pierdute sau de intervale de ceas reduse nu poate face ca protocolul să furnizeze pachete duplicate către vreunul dintre nivelurile rețea, să sară un pachet sau să se blocheze.

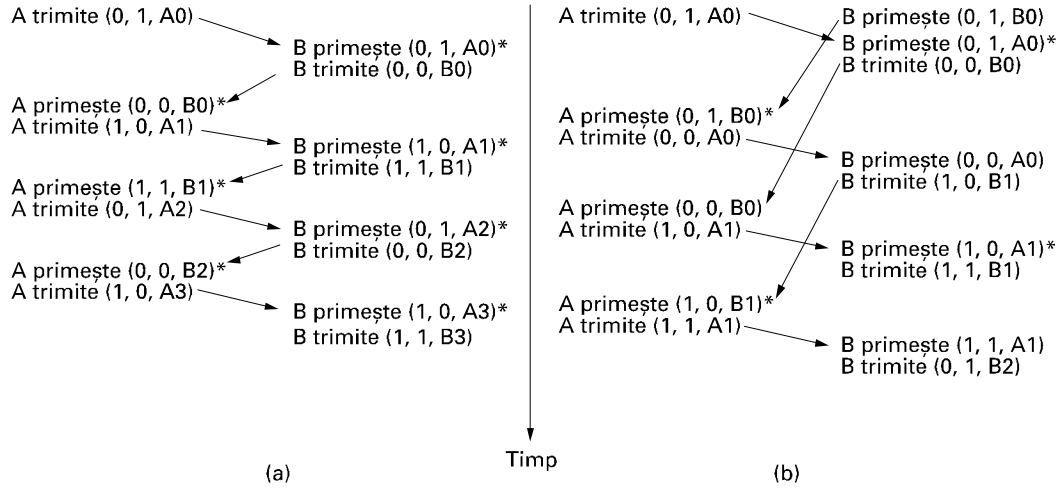

A trimite (0, 1, A0)
B primește (0, 1, A0)*
B trimite (0, 0, B0)
A primește (0, 0, B0)*
A trimite (1, 0, A1)
B primește (1, 0, A1)*
B trimite (1, 1, B1)
A primește (1, 1, B1)*
A trimite (0, 1, A2)
B primește (0, 1, A2)*
B trimite (0, 0, B2)
A primește (0, 0, B2)*
A trimite (1, 0, A3)
B primește (1, 0, A3)*
B trimite (1, 1, B3)
(a)
Timp
A trimite (0, 1, A0)
B primește (0, 1, B0)
B primește (0, 1, A0)*
B trimite (0, 0, B0)
A primește (0, 1, B0)*
A trimite (0, 0, A0)
B primește (0, 0, A0)
B trimite (1, 0, B1)
A primește (0, 0, B0)
A trimite (1, 0, A1)
B primește (1, 0, A1)*
B trimite (1, 1, B1)
A primește (1, 0, B1)*
A trimite (1, 1, A1)
B primește (1, 1, A1)
B trimite (0, 1, B2)
(b)

Fig. 3-15. Două scenarii pentru protocolul 4. (a) Cazul normal. (b) Cazul special. Notația este (seq, ack, packet number). Un asterisc arată că nivelul rețea acceptă un pachet.

Totuși, dacă ambele părți trimit simultan un pachet inițial, atunci apare o situație specială. Această dificultate de sincronizare este ilustrată de fig. 3-15. În partea (a) este prezentată funcționarea normală a protocolului. În (b) este ilustrată situația specială. Dacă *B* așteaptă primul cadru de la *A* înainte de a-l trimite pe al său, secvența de acțiuni este cea arătată în (a) și fiecare cadru este acceptat. Totuși, dacă *A* și *B* inițiază comunicația simultan, primele lor cadre se încrucișează și nivelurile legătură de date ajung în situația (b). În (a) fiecare sosire de cadru aduce un nou pachet pentru nivelul rețea; nu există duplicate. În (b) jumătate din cadre conțin duplicate, chiar dacă nu există erori de transmisie. Situații similare pot să apară ca rezultat al expirării premature a timpului, chiar dacă una dintre părți începe prima. De fapt, dacă intervin mai multe expirări premature, atunci cadrele pot fi trimise de trei sau mai multe ori.

3.4.2 Un protocol de revenire cu *n* pași (Go Back n)

Până acum am făcut presupunerea tacită că timpul de transmisie necesar pentru ca un cadru să ajungă la receptor plus timpul de transmisie a confirmării este neglijabil. Uneori această presupunere este în mod cert falsă. În aceste situații timpul mare de transfer poate avea implicații importante pentru eficiența utilizării lărgimii de bandă. Ca exemplu, să considerăm un canal de satelit de 50 Kbps cu timpul de întârziere datorită propagării dus-întors de 500 milisecunde. Să ne imaginăm că încercăm să utilizăm protocolul 4 pentru a trimite cadre de 1000 de biți prin satelit. La $t = 0$ emițătorul începe să trimită primul cadru. Considerând cele mai optimiste condiții (fără așteptare la receptor și un cadru de confirmare scurt), cadrul nu poate ajunge în totalitate la receptor înainte de $t = 270$ milisecunde, iar confirmarea nu poate ajunge înapoi la emițător înainte de $t = 520$ milisecunde. Aceasta înseamnă că emițătorul a fost blocat pentru 500/520 sau 96% din timp. Cu alte cuvinte, a fost utilizată doar 4% din lărgimea de bandă. Evident, combinația dintre un timp de tranziție lung, lărgime de bandă mare și un cadru de lungime mică este dezastruoasă din punct de vedere al eficienței.

Problema descrisă anterior poate fi privită ca o consecință a regulii care cere ca un emițător să aștepte o confirmare înaintea trimiterii unui alt cadru. Dacă relaxăm această restricție, poate fi atinsă o eficiență mult mai ridicată. Practic, soluția constă în a permite emițătorului să transmită până la *w* cadre, în loc de unul singur. Cu o alegere potrivită a lui *w* emițătorul va putea să transmită continuu cadre pentru un timp egal cu timpul de tranzit, fără a umple fereastra. În exemplul anterior *w* va fi minim 26. Emițătorul începe emiterea cadrului 0 ca mai înainte. În momentul în care se termină trimiterea a 26 de cadre, la $t = 520$, va sosi și confirmarea pentru cadrul 0. Apoi, confirmările vor sosi la fiecare 20 milisecunde, așa încât emițătorul primește întotdeauna permisiunea să continue exact atunci când dorește. În permanență există 25 sau 26 cadre neconfirmate. Cu alte cuvinte dimensiunea maximă a ferestrei emițătorului este de 26.

Nevoia pentru o fereastră mare la emițător apare atunci când produsul lărgime de bandă x timpul de propagare dus-întors este mare. Dacă lărgimea de bandă este mare, chiar și pentru întârzieri moderate, emițătorul își va termina repede fereastra. Daca întârzierea este mare (de exemplu, canal de satelit), emițătorul își va termina fereastra chiar și pentru lărgimi de bandă moderate. Produsul acestor doi factori spune de fapt care este capacitatea canalului, iar pentru a opera la eficiență maximă, emițătorul trebuie sa fie capabil sa o umple fără să se oprească.

Această tehnică este cunoscută ca **bandă de asamblare** (eng.: **pipelining**). Considerând capacitatea canalului de *b* biți pe secundă, dimensiunea cadrului de *l* biți și timpul de propagare dus-întors *R* secunde, timpul necesar pentru a transmite un singur cadru este *l/b* secunde. După ce a fost transmis ultimul bit al unui cadru de date, apare o întârziere de *R/2* înainte ca biții să ajungă la receptor și o altă întârziere de cel puțin *R/2* pentru sosirea confirmării, rezultând o întârziere totală de *R*. În cazul protocoalelor pas-cu-pas, linia este ocupată pentru un timp egal cu *l/b* și în așteptare pentru un timp egal cu *R*, rezultând:

$$\text{utilizarea liniei} = l/(l+bR).$$

Dacă $l < bR$, eficiența va fi mai mică de 50%. Deoarece până la întoarcerea confirmării există întotdeauna o întârziere nenulă, în principiu poate fi folosită banda de asamblare, pentru a ține linia ocupată tot acest interval, dar dacă intervalul este mic, complexitatea suplimentară face efortul inutil.

Utilizarea benzii de asamblare în cazul unui canal de comunicație nesigur ridică probleme serioase. Mai întâi să vedem ce se întâmplă dacă un cadru din mijlocul unui șir lung este modificat sau pierdut. Multe cadre succesive vor ajunge la receptor înainte ca emițătorul să observe că ceva

este greșit. Atunci când un cadru modificat ajunge la receptor este evident că el trebuie eliminat, dar ce trebuie să facă receptorul cu toate cadrele corecte care urmează? Să reamintim că nivelul legătură de date receptor este obligat să livreze pachete către nivelul rețea în secvență. În fig. 3-16, se prezintă efectele utilizării benzii de asamblare asupra revenirii în caz de eroare. Acum le vom examina în detaliu.

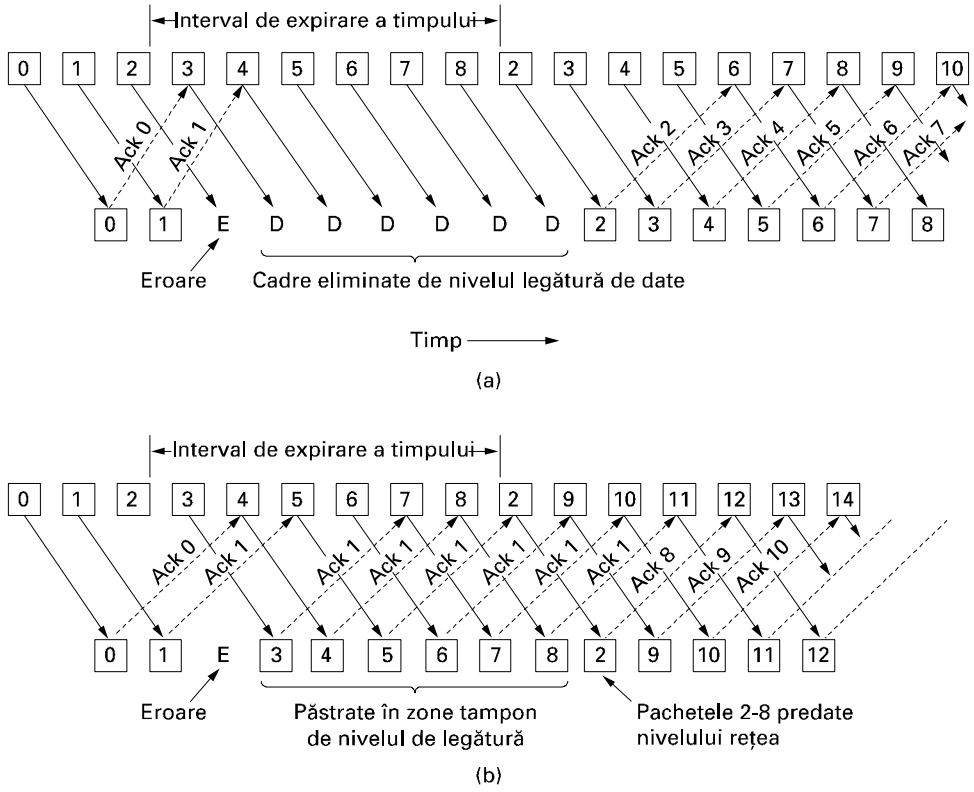

Interval de expirare a timpului
0 1 2 3 4 5 6 7 8 2 3 4 5 6 7 8 9 10
Ack 0
Ack 1
Ack 2
Ack 3
Ack 4
Ack 5
Ack 6
Ack 7
0 1 E D D D D D D 2 3 4 5 6 7 8
Eroare
Cadre eliminate de nivelul legătură de date
Timp
(a)
Interval de expirare a timpului
0 1 2 3 4 5 6 7 8 2 9 10 11 12 13 14
Ack 0
Ack 1
Ack 1
Ack 1
Ack 1
Ack 1
Ack 1
Ack 1
Ack 8
Ack 9
Ack 10
0 1 E 3 4 5 6 7 8 2 9 10 11 12
Eroare
Păstrate în zone tampon de nivelul de legătură
Pachetele 2-8 predate nivelului rețea
(b)

Fig. 3-16. Folosirea benzii de asamblare și revenirea din eroare. Efectul unei erori când (a) dimensiunea ferestrei receptoare este 1 și (b) dimensiunea ferestrei receptorului este mare.

Există două moduri de bază de tratare a erorilor în prezența benzii de asamblare. Un mod, numit **revenire cu n pași** (eng.: **go back n**), este ca receptorul să elimine pur și simplu cadrele care urmează, netrimițând confirmări pentru cadrele eliminate. Această strategie corespunde unei ferestre de recepție de dimensiune 1. Cu alte cuvinte, nivelul legătură de date refuză să accepte orice cadru exceptându-l pe următorul care trebuie livrat către nivelul rețea. Dacă fereastra emițătorului se umple înaintea expirării contorului de timp, banda de asamblare va începe să se golească. În cele din urmă, timpul emițătorului va expira și se vor retransmite toate cadrele neconfirmate, în ordine, începând cu cadrul pierdut sau modificat. Dacă rata erorilor este mare, această abordare poate risipi o mare parte din lărgimea de bandă.

În fig. 3-16 (a) este prezentat protocolul de revenire cu n pași pentru cazul în care fereastra receptorului are dimensiune unu. Cadrele 0 și 1 sunt primite și confirmate corect. Cadrul 2, totuși, este alterat sau pierdut. Emițătorul, care nu știe de această problemă, continuă să trimită cadre până

când timpul pentru cadrul 2 expiră. Apoi se întoarce la cadrul 2 și o ia de la început cu el, trimițând din nou cadrele 2, 3, 4 etc.

Cealaltă strategie generală de tratare a erorilor atunci când este folosită banda de asamblare se numește **repetare selectivă** (eng.: **selective repeat**). Când aceasta este utilizată, un cadru incorect este respins, dar toate cadrele corecte care îl urmează sunt memorate. Când contorul de timp al emițătorului expiră, cel mai vechi cadru neconfirmat este retransmis. Dacă acest cadru ajunge corect, receptorul poate transmite către nivelul rețea, în ordine, cadrele pe care le-a memorat. Repetarea selectivă este deseori combinată cu utilizarea confirmărilor negative (NAK), care sunt trimise atunci când se detectează o eroare, de exemplu când se primește un cadru cu suma de control incorectă sau cu număr de secvență necorespunzător. Confirmările negative simulează retransmisia înainte de expirarea contorului de timp corespunzător, îmbunătățind astfel performanța.

În fig. 3-16 (b), cadrele 0 și 1 sunt recepționate corect, dar confirmarea pentru cadrul 2 este pierdută. Când cadrul 3 sosește la receptor, nivelul legătură de date observă că a pierdut un cadru, și trimite o confirmare negativă pentru 2, memorând însă cadrul primit. Când cadrele 4, 5 ajung la receptor, sunt la rândul lor memorate de nivelul legătură de date, în loc de a fi transmise nivelului rețea. În cele din urmă, confirmarea negativă pentru 2 ajunge înapoi la emițător, care va retransmite cadrul 2. Când acesta ajunge la receptor, nivelul legătură de date are cadrele 2, 3, 4, 5, și le poate pasa nivelului rețea în ordinea corectă. De asemenea, poate confirma toate cadrele până la 5 inclusiv, așa cum se prezintă în figură. Dacă NAK-ul se pierde, în cele din urmă contorul de timp al emițătorului va expira și acesta va iniția retransmisia, dar în acest fel se pierde mai mult timp. În concluzie, utilizarea confirmărilor negative accelerează retransmiterea unui anumit cadru.

Strategia de repetare selectivă corespunde unei ferestre a receptorului mai mare ca 1. Orice cadru din interiorul ferestrei poate fi acceptat și memorat până când toate cele precedente vor fi trimise nivelului rețea. Dacă fereastra este mare, această abordare poate necesita un spațiu mare de memorie pentru nivelul legătură de date.

Aceste două alternative reprezintă compromisuri între lărgimea de bandă și spațiul ocupat de tampoane la nivelul legătură de date. În funcție de care resursă este mai deficitară, poate fi utilizată una sau cealaltă. Fig. 3-17 prezintă un protocol de tip bandă de asamblare în care nivelul legătură de date receptor acceptă cadre ordonate; cadrele ce urmează după o eroare sunt eliminate. În acest protocol, pentru prima dată, am renunțat la presupunerea că nivelul rețea are o rezervă infinită de pachete care trebuie trimise. Atunci când nivelul rețea are un pachet pe care dorește să-l trimită, poate produce un eveniment *network_layer_ready*. Totuși, pentru a impune regula de control al fluxului, conform căreia nu pot exista decât cel mult *MAX_SEQ* cadre neconfirmate, nivelul legătură de date trebuie să poată să interzică nivelului rețea să îl perturbe cu mai multe. Această funcție este realizată de funcțiile de bibliotecă *enable_network_layer* și *disable_network_layer*.

Observați că în orice moment pot exista cel mult *MAX_SEQ* cadre și nu *MAX_SEQ*+1 cadre neconfirmate, chiar dacă există *MAX_SEQ*+1 numere de secvență: 0, 1, 2, ...*MAX_SEQ*. Pentru a vedea de ce este necesară această restricție, să considerăm următorul scenariu cu *MAX_SEQ* = 7.

1. Emițătorul trimite cadrele de la 0 la 7.
2. O confirmare atașată pentru cadrul 7 ajunge la emițător.
3. Emițătorul trimite alte opt cadre, din nou cu numerele de secvență de la 0 la 7.
4. Acum ajunge o altă confirmare atașată pentru cadrul 7.

```
/* Protocolul 5 (revenire cu n pași) permite mai multe cadre în așteptare. Emițătorul poate
trimite până la MAX_SEQ cadre fără a aștepta confirmare. În plus, spre deosebire de proto-
coalele precedente, acesta nu presupune că nivelul rețea ar avea tot timpul un nou pachet.
În schimb, nivelul rețea provoacă un eveniment network_layer_ready atunci când are de tri-
mis un pachet */

#define MAX_SEQ 7                                              /* trebuie să fie 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
 /* întoarce adevărat dacă a <= b < c în mod circular și fals în caz contrar */
 if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
   return(true);
 else
   return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
 /* construiește și trimite un cadru de date */
 frame s;                                                      /* variabilă temporară */

 s.info = buffer[frame_nr];                                /* inserează pachetul în cadru */
 s.seq = frame_nr;                                  /* inserează numărul de secvență în cadru */
 s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);             /* atașează confirmarea */
 to_physical_layer(&s);                                          /* transmite cadrul */
 start_timer(frame_nr);                                          /* pornește ceasul */
}

void protocol5(void)
{
 seq_nr next_frame_to_send;              /* MAX_SEQ > 1; utilizat pentru fluxul de ieșire */
 seq_nr ack_expected;                          /* cel mai vechi cadru încă neconfirmat */
 seq_nr frame_expected;                /* următorul cadru așteptat, din fluxul de intrare */
 frame r;                                                      /* variabilă auxiliară */
 packet buffer[MAX_SEQ+1];                          /* zone tampon pentru fluxul de ieșire */
 seq_nr nbuffered;                 /* număr de zone tampon de ieșire utilizate în prezent */
 seq_nr i;                                /* utilizat ca index în vectorul de zone tampon */
 event_type event;
 enable_network_layer();                        /* permite evenimente network_layer_ready */
 ack_expected = 0;                                    /* următoarea confirmare așteptată */
 next_frame_to_send = 0;                                    /* următorul cadru transmis */
 frame_expected = 0;                             /* numărul cadrului așteptat să sosească */
 nbuffered = 0;                       /* inițial în zonele tampon nu există nici un pachet */
 while(true) {
   wait_for_event(&event);               /* patru posibilități: vezi event_type, mai sus */
   switch(event) {
     case network_layer_ready:                   /* nivelul rețea are un pachet de trimis */
       /* acceptă, salvează și transmite un nou cadru */
       from_network_layer(&buffer[next_frame_to_send]);             /* preia noul pachet */
       nbuffered = nbuffered + 1;                       /* extinde fereastra emițătorului */
       send_data(next_frame_to_send, frame_expected, buffer);        /* transmite cadrul */
       inc(next_frame_to_send);     /* crește limita superioară a ferestrei emițătorului */
       break;
```

```
        case frame_arrival:                      /* a sosit un cadru de date sau de control */
          from_physical_layer(&r);               /* preia de la nivelul fizic cadrul sosit */

          if (r.seq == frame_expected) {
                                                 /* cadrele sunt acceptate doar în ordine */
          to_network_layer(&r.info);                     /* predă pachetul nivelului rețea */
                    inc(frame_expected);/* crește limita inferioară a ferestrei emițătorului*/
          }
          /* Confirmarea lui n implică n-1, n-2 etc. Verifică acest lucru. */
          while (between(ack_expected, r.ack, next_frame_to_send)) {
          /* tratează confirmarea atașată */
          nbuffered = nbuffered - 1;               /* un cadru mai puțin în zonele tampon */
          stop_timer(ack_expected);               /* cadrul a sosit intact; oprește ceasul */
          inc(ack_expected);                           /* contractă fereastra emițătorului */
          }
          break;
        case cksum_err: break;                /* cadrele eronate sunt pur și simplu ignorate */
        case timeout:                        /* necaz; retransmite toate cadrele neconfirmate */
          next_frame_to_send = ack_expected;              /* începe retransmiterea de aici */
          for (i=1; i <= nbuffered; i++) {
               send_data(next_frame_to_send, frame_expected, buffer);/* retransmite 1 cadru */
            inc(next_frame_to_send);               /* pregătește transmiterea următorului */
          }

      }

      if (nbuffered < MAX_SEQ)
        enable_network_layer();
      else
        disable_network_layer();
   }
}
```

Fig. 3-17. Un protocol cu fereastră glisantă utilizând revenirea cu n pași.

Întrebarea este: toate cele opt cadre aparținând celui de al doilea lot au ajuns corect ori s-au pierdut în totalitate (considerând respingerile care urmează unei erori ca pierderi)? În ambele cazuri receptorul va trimite cadrul 7 ca o confirmare, deci emițătorul nu are posibilitatea să știe. Din acest motiv, numărul maxim de cadre neconfirmate trebuie limitat la *MAX_SEQ*.

Chiar dacă protocolul 5 nu păstrează cadrele sosite după o eroare, problema memorării nu dispare. Deoarece un emițător poate avea de retransmis la un moment de timp viitor toate cadrele neconfirmate, el trebuie să păstreze toate cadrele transmise până când va fi sigur că au fost acceptate de receptor. Când sosește o confirmare pentru cadrul *n*, cadrele *n*-1, *n*-2 ș.a.m.d. sunt confirmate automat. Această proprietate este foarte importantă atunci când unele dintre cadrele purtătoare de confirmări au fost pierdute sau modificate. De fiecare dată, când sosește o confirmare, nivelul legătură de date verifică să vadă dacă unele tampoane pot fi eliberate. Dacă tampoanele pot fi eliberate (adică există spațiu disponibil în fereastră), atunci nivelul rețea anterior blocat poate primi permisiunea să producă alte evenimente *network_layer_ready*.

În cazul acestui protocol, presupunem că există întotdeauna trafic în direcția inversă de care să se poată atașa confirmările. În caz contrar, confirmările nu pot fi trimise. Protocolul 4 nu are nevoie de această presupunere deoarece acesta trimite înapoi un cadru de fiecare dată când recepționează unul. În protocolul următor, vom rezolva această problemă într-o manieră elegantă.

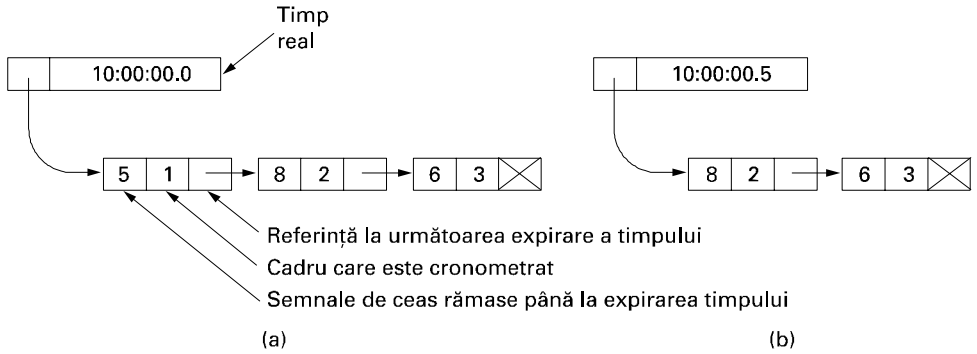


Fig. 3-18. Simularea prin program a contoarelor de timp multiple.

Deoarece protocolul 5 are mai multe cadre neconfirmate, este evident că necesită mai multe contoare de timp, câte unul pentru fiecare cadru neconfirmat. Timpul fiecărui cadru expiră independent de toate celelalte. Toate aceste contoare pot fi simulate ușor prin program, utilizând un singur ceas fizic care produce întreruperi periodice. Contoarele de timp active formează o listă înlănțuită, în fiecare nod existând informații despre câte semnale de ceas mai sunt până la expirarea timpului, cadrul care este cronometrat și un pointer către următorul nod.

Ca o ilustrare a modului în care pot fi implementate contoarele de timp, să considerăm exemplul din fig. 3-18(a). Să presupunem că impulsurile de ceas sunt la fiecare 100 ms. Inițial timpul real este 10:00:00.0; există trei timpi de expirare, la 10:00:00.5, 10:00:01.3 și 10:00:01.9. La fiecare impuls de ceas produs de echipament, timpul real este actualizat și contorul de impulsuri de la capătul listei este decrementat. Atunci când contorul de impulsuri de ceas devine zero, se produce o expirare de timp și nodul este scos din listă, ca în fig. 3-18 (b). Chiar dacă această organizare cere ca lista să fie parcursă când este apelat *start_timer* sau *stop_ timer*, nu va necesita multe operații la fiecare impuls de ceas. În protocolul 5, ambele rutine au un parametru indicând pentru ce cadru se face contorizarea timpului.

3.4.3 Un protocol cu repetare selectivă

Protocolul 5 funcționează bine dacă erorile sunt rare, dar dacă linia este slabă, se pierde mult din lărgimea de bandă cu retransmiterea cadrelor. O altă strategie de tratare a erorilor este ca receptorul să accepte și să numeroteze cadrele care urmează după un cadru deteriorat sau pierdut. Un astfel de protocol nu elimină cadre doar pentru că un cadru anterior a fost deteriorat sau pierdut.

În acest protocol, atât emițătorul cât și receptorul mențin o fereastră de numere de secvență acceptabile. Dimensiunea ferestrei emițătorului începe de la 0 și crește până la un maxim predefinit *MAX_SEQ*. Spre deosebire de aceasta, fereastra receptorului are dimensiunea fixă *MAX_SEQ*. Receptorul are un tampon rezervat pentru fiecare număr de secvență din cadrul ferestrei. Fiecare tampon are un bit asociat (*arrived - sosit*) care ne spune dacă tamponul este plin sau gol. De fiecare dată când sosește un cadru, numărul său de secvență este verificat de funcția *between,* pentru a vedea dacă face parte din fereastră. Dacă da, și dacă nu a fost deja recepționat, este acceptat și memorat. Această acțiune are loc fără să se verifice dacă conține sau nu următorul pachet așteptat de nivelul rețea. Desigur cadrul trebuie păstrat la nivelul legătură de date și nu trebuie trimis către nivelul rețea decât atunci când toate cadrele cu numere mai mici au fost deja livrate nivelului rețea în ordinea corectă. Un protocol utilizând acest algoritm este prezentat în fig. 3 -19.

```
/* Protocolul 6 (repetare selectivă) acceptă cadrele în afara secvenței, dar predă pachetele în ordine nivelului rețea. Fiecărui cadru neconfirmat îi este asociat un ceas. La expirarea timpului este retransmis doar acest cadru și nu toate cele neconfirmate, ca în protocolul 5. */

#define MAX_SEQ 7                                        /* trebuie să fie 2^n - 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
  event_type;
#include "protocol.h"
boolean no_nak = true;              /* încă nu a fost trimisă nici o confirmare negativă */
seq_nr oldest_frame = MAX_SEQ + 1;
static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* La fel ca between din protocolul 5, dar mai scurt și mai neclar. */
 return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}
static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet
buffer[])
{
 /* construiește și trimite un cadru de date, de ack sau de nak */
 frame s;                                                    /* variabilă temporară */

 s.kind = fk;                                             /* kind == data, ack sau nak */
 if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
 s.seq = frame_nr;                                /* are sens doar pentru cadrele de date */
 s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
 if (fk == nak) no_nak = false;                              /* un nak per cadru, te rog */
 to_physical_layer(&s);                                          /* transmite cadrul */
 if (fk == data) start_timer(frame_nr % NR_BUFS);
 stop_ack_timer();                           /* nu este nevoie de un cadru de ack separat */
}
void protocol6(void)
{
 seq_nr ack_expected;                      /* limita inferioară a ferestrei emițătorului */
 seq_nr next_frame_to_send;              /* limita superioară a ferestrei emițătorului + 1*/
 seq_nr frame_expected;                    /* limita inferioară a ferestrei receptorului */
 seq_nr too_far;                         /* limita superioară a ferestrei receptorului + 1*/
 int i;                                                    /* indicele zonei tampon */
 frame r;                                                    /* variabilă temporară */
 packet out_buf[NR_BUFS];                          /* zone tampon pentru fluxul de ieșire */
 packet in_buf[NR_BUFS];                           /* zone tampon pentru fluxul de intrare */
 boolean arrived[NR_BUFS];                                 /* hartă de biți de intrare */
 seq_nr nbuffered;                /* câte zone tampon de ieșire sunt folosite în prezent */
 event_type event;
 enable_network_layer();                                          /* inițializează */
 ack_expected = 0;               /* următoarea confirmare așteptată în fluxul de intrare */
 next_frame_to_send = 0;                            /* numărul următorului cadru transmis */
 frame_expected = 0;
 too_far = NR_BUFS;
 nbuffered = 0;                          /* inițial zonele tampon nu conțin nici un pachet */
for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
while(true) {
 wait_for_event();                           /* cinci variante: vezi event_type, mai sus */
```

```
switch(event) {
  case network_layer_ready:                /* acceptă, salvează și trimite un nou cadru */
    nbuffered = nbuffered + 1;                                        /* extinde fereastra */
    from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* preia un nou pachet */
    send_frame(data, next_frame_to_send, frame_expected, out_buf); /* trimite cadrul */
    inc(next_frame_to_send);                 /* avansează limita superioară a ferestrei */
    break;
  case frame_arrival:                        /* a sosit un cadru de date sau de control */
    from_physical_layer(&r);                 /* preia cadrul sosit de la nivelul fizic */
    if (r.kind == data) {
      /* A sosit un cadru nedeteriorat */
      if ((r.seq != frame_expected) && no_nak)
        send_frame(nak, 0, frame_expected, out_buf);
  else start_ack_timer();
      if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS]==false)) {
        /* Cadrele pot fi acceptate în orice ordine */
        arrived[r.seq%NR_BUFS] = true;              /* marchează tamponul ca fiind plin */
        in_buf[r.seq%NR_BUFS] = r.info;                  /* introduce datele în tampon */
        while (arrived[frame_expected % NR_BUFS]) {
        /* Predă cadrele și avansează fereastra */
        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
        no_nak = true;
        arrived[frame_expected % NR_BUFS] = false;
        inc(frame_expected);     /* avansează limita inferioară a ferestrei receptorului */
        inc(too_far);            /* avansează limita superioară a ferestrei receptorului */
        start_ack_timer();              /* pentru a stabili dacă e necesar ack separat */
        }
      }
    }
    if ((r.kind==nak)&&between(ack_expected, (r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
      send_frame(data, (r.ack+1)%(MAX_SEQ+1), frame_expected, out_buf);
    while (between(ack_expected, r.ack, next_frame_to_send)) {
      nbuffered = nbuffered - 1;                                   /* tratează ack atașat */
      stop_timer(ack_expected % NR_BUFS);                        /* cadrul a ajuns intact */
      inc(ack_expected);       /* avansează marginea inferioară a ferestrei emițătorului */
    }
    break;
  case cksum_err:
    if (no_nak) send_frame(nak, 0, frame_expected, out_buf);         /* cadru deteriorat */
    break;
  case timeout:
    send_frame(data, oldest_frame, frame_expected, out_buf);         /* a expirat timpul */
    break;
  case ack_timeout:
    send_frame(ack, 0, frame_expected, out_buf);                          /* timpul asociat
                                          confirmării pozitive a expirat; trimite ack */
  }
  if (nbuffered ( NR_BUFS) enable_network_layer();
 else disable_network_layer();
}
}
```

Fig. 3-19. Un protocol cu fereastră glisantă utilizând repetarea selectivă.

Recepția nesecvențială introduce anumite probleme ce nu sunt prezente în protocoalele în care cadrele sunt recepționate numai în ordine. Putem ilustra problemele foarte ușor cu un exemplu. Să presupunem că avem un număr de secvență pe trei biți și deci emițătorul poate transmite până la șapte cadre înainte să fie necesar să aștepte o confirmare. Inițial ferestrele emițătorului și receptorului arată ca în fig. 3-20(a). Emițătorul trimite acum cadrele de la 0 la 6. Fereastra receptorului îi permite să accepte orice cadru cu număr de secvență între 0 și 6 inclusiv. Toate cele șapte cadre sosesc corect, deci receptorul le confirmă avansându-și fereastra pentru a permite recepția cadrelor 7, 0, 1, 2, 3, 4 sau 5, așa cum arată fig. 3-20 (b). Toate cele 7 tampoane sunt marcate ca fiind goale.

În acest punct se produce dezastrul, din cauza unui fulger care lovește linia telefonică, înlăturând toate confirmările. Emițătorul ajunge în cele din urmă la timeout și retransmite cadrul 0. Atunci când acest cadru sosește la receptor, este făcută o verificare pentru a vedea dacă se încadrează în fereastra receptorului. Din păcate, în fig. 3-20(b) cadrul 0 este în interiorul noii ferestre și deci va fi acceptat. Receptorul trimite o confirmare atașată pentru cadrul 6, deoarece au fost recepționate cadrele de la 0 la 6.

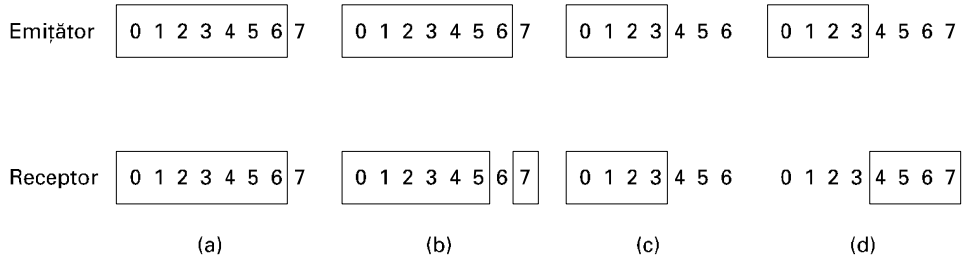


Fig. 3-20. (a) Situația inițială cu o fereastră de dimensiune 7. (b) După ce șapte cadre au fost trimise și recepționate, dar neconfirmate. (c) Situația inițială cu o fereastră de dimensiune patru. (d) După ce patru cadre au fost trimise și recepționate, dar neconfirmate.

Emițătorul este fericit să afle că toate cadrele transmise au ajuns corect, deci își avansează fereastra și trimite apoi imediat cadrele 7, 0, 1, 2, 3, 4 și 5. Cadrul 7 va fi acceptat de receptor și pachetul său va fi trimis direct nivelului rețea. Imediat după aceea, nivelul legătură de date receptor verifică să vadă dacă are un cadru 0 corect, descoperă că îl are și trimite pachetul conținut în el nivelului rețea. În consecință, nivelul rețea primește un pachet incorect și protocolul eșuează.

Esența problemei este aceea că după ce receptorul își avansează fereastra, noul interval de numere de secvență corecte se suprapune peste cel vechi. Ca urmare, următoarea serie de cadre pot să fie sau duplicate (dacă toate confirmările sunt pierdute) sau unele noi (dacă toate confirmările au fost recepționate). Receptorul nu are nici o posibilitate de a distinge cele două cazuri.

Pentru a ieși din această dilemă, trebuie să ne asigurăm că atunci când receptorul își deplasează fereastra, nu există nici o suprapunere peste cea anterioară. Pentru a asigura că nu există suprapunere, dimensiunea maximă a ferestrei trebuie să fie de cel mult jumătate din intervalul numerelor de secvență, așa cum se face în fig. 3-20 (c) și fig. 3-20 (d). De exemplu, dacă pentru numerele de secvență sunt utilizați 4 biți, acestea vor lua valori de la 0 la 15. În orice moment trebuie să existe numai opt cadre. Astfel, dacă receptorul tocmai a acceptat cadrele de la 0 la 7 și avansează fereastra pentru a permite acceptarea cadrelor de la 8 la 15, poate spune cu certitudine dacă următoarele cadre sunt retransmisii (de la 0 la 7) sau sunt unele noi (de la 8 la 15). În general, dimensiunea ferestrei pentru protocolul 6 va fi $(MAX\_SEQ+1)/2$. Deci, pentru numere de secvență pe 4 biți, dimensiunea ferestrei este 4.

O întrebare interesantă este: câte tampoane trebuie să aibă receptorul? În nici un caz el nu va accepta cadre ale căror numere de secvență sunt sub limita minimă a ferestrei, sau cadre ale căror numere de secvență depășesc limita maximă a acesteia. În consecință, numărul de tampoane necesare este egal cu dimensiunea ferestrei, nu cu intervalul de valori al numerelor de secvență. În exemplul anterior, cu numere de secvență pe 4 biți, sunt necesare 8 tampoane, numerotate de la 0 la 7. Atunci când sosește cadrul *i*, acesta este pus în tamponul *i* mod 8. De notat că, deși *i* și (*i*+8) mod 8 „concurează" pentru același tampon, nu vor fi în aceeași fereastră simultan, deoarece aceasta ar implica o dimensiune a ferestrei de cel puțin 9.

Pentru același motiv, numărul de contoare de timp necesare este egal cu numărul de tampoane, nu cu dimensiunea spațiului secvențelor. Efectiv, există un contor de timp asociat fiecărui tampon. Atunci când contorul expiră, conținutul tamponului este retransmis.

În protocolul 5, s-a presupus în mod implicit că acel canal este puternic încărcat. Când sosește un cadru, nu se trimite imediat o confirmare. Confirmarea este atașată la următorul cadru de date de ieșire. Dacă traficul invers este slab, confirmarea va fi reținută o perioadă mare de timp. Dacă traficul este intens într-o direcție și inexistent în cealaltă direcție, atunci sunt trimise numai *MAX_SEQ* cadre și apoi protocolul se blochează, de aceea am presupus că exista întotdeauna trafic în direcția inversă.

Această problemă este rezolvată în protocolul 6. După sosirea unei secvențe de cadre cu date, este pornit un contor de timp auxiliar, prin *start_ack_timer*. Dacă până la expirarea acestui contor nu a apărut trafic în sens invers, atunci este trimis un cadru de confirmare separat. O întrerupere datorată contorului auxiliar se numește eveniment *ack_timeout*. Cu acest artificiu, fluxul de trafic unidirecțional este acum posibil, deoarece absența cadrelor de date în sens invers, pe care pot fi atașate confirmări, nu mai este un obstacol. Există numai un contor auxiliar și dacă *start_ack_timer* este apelată în timpul funcționării contorului, acesta este resetat la un interval complet de timp de confirmare.

Este esențial ca timpul de expirare asociat contorului auxiliar să fie mult mai scurt decât cel utilizat pentru cadrele de date de ieșire. Această condiție este impusă pentru a ne asigura că o confirmare pentru un cadru corect recepționat sosește înainte ca timpul emițătorului să expire și acesta să retransmită cadrul.

Protocolul 6 utilizează pentru tratarea erorilor o strategie mai eficientă decât protocolul 5. De fiecare dată când receptorul are motiv să suspecteze că a apărut o eroare, trimite înapoi la emițător un cadru cu o confirmare negativă (NAK). Un asemenea cadru reprezintă o cerere pentru retransmiterea cadrului specificat în NAK. Există două cazuri în care receptorul ar trebui să fie suspicios: a sosit un cadru modificat sau a sosit un alt cadru decât cel așteptat (un posibil cadru pierdut). Pentru a preveni producerea cererilor multiple de retransmisie a aceluiași cadru pierdut, receptorul va ține minte dacă un NAK a fost deja trimis pentru un anumit cadru. Variabila *no_nak* din protocolul 6 are valoarea adevărat dacă nici un NAK nu a fost trimis pentru *frame_expected*. Dacă NAK a fost modificat sau pierdut, nu se întâmplă nimic, deoarece emițătorul va ajunge, până la urmă, la timeout și va retransmite cadrul lipsă. Dacă un cadru greșit sosește după ce un NAK a fost transmis și pierdut, *no_nak* va fi adevărat și va fi pornit contorul de timp auxiliar. La expirarea acestuia, va fi trimis un ACK pentru resincronizarea emițătorului cu starea curentă a receptorului.

În unele situații, timpul necesar pentru ca un cadru să se propage la destinație, să fie prelucrat și să se recepționeze confirmarea este (aproape) constant. În aceste condiții, emițătorul își poate ajusta contorul de timp să fie puțin mai mare decât intervalul de timp normal așteptat între emiterea unui cadru și recepționarea confirmării sale. Totuși, dacă acest timp variază puternic, emițătorul trebuie să aleagă între fixarea intervalului la o valoare mică (riscând retransmisii inutile) și fixarea la o valoare mare (rămânând în așteptare timp îndelungat după producerea unei erori).

În ambele cazuri se irosește lărgime de bandă. Dacă traficul în sens invers este sporadic, timpul dinaintea confirmării va avea valori neregulate, fiind scurt când există trafic în sens invers și lung când nu există. Variația timpului de prelucrare la receptor poate fi, de asemenea, o problemă. În general, atunci când deviația standard a intervalului de confirmare este mică, în comparație cu intervalul însuși, intervalul de timp poate fi „strâmt" și NAK-urile nu sunt utile. Altfel, contorul de timp trebuie să fie setat "larg" și NAK-urile pot accelera apreciabil retransmisia cadrelor eronate sau pierdute.

Strâns legată de problema expirării timpului și NAK-urilor este problema determinării cadrului care a cauzat expirarea timpului. În protocolul 5 acesta este întotdeauna *ack_expected*, deoarece este întotdeauna cel mai vechi. În protocolul 6 nu este ușor să se determine cel care a produs expirarea timpului. Să presupunem că au fost transmise cadrele de la 0 la 4, însemnând că lista cadrelor neconfirmate este 0, 1, 2, 3, 4, în ordinea de la cel mai vechi la cel mai nou. Acum să ne imaginăm că expiră timpul pentru 0, este transmis 5 (un nou cadru), expiră timpul pentru 1, expiră timpul pentru 2 și este transmis 6 (un alt cadru nou). În acest moment, lista cadrelor neconfirmate este 3, 4, 0, 5, 1, 2, 6, de la cel mai vechi la cel mai nou. Dacă tot traficul de răspuns (mai precis, cadrele cu confirmări) este pierdut pentru un timp, expirarea timpului pentru cele șapte cadre neconfirmate se va produce în această ordine. Pentru a nu face ca exemplul să fie mai complicat decât este deja, nu am prezentat administrarea contoarelor de timp. În schimb, am presupus că, la expirarea timpului, variabila *oldest_frame* este setată astfel, încât să indice cadrul pentru care a trecut timpul.

3.5 VERIFICAREA PROTOCOALELOR

Protocoalele reale și programele ce le implementează sunt adesea destul de complicate. Ca urmare, a fost întreprinsă o imensă muncă de cercetare pentru a găsi tehnici formale, matematice, pentru specificarea și verificarea protocoalelor. În secțiunile următoare vom studia câteva astfel de modele și tehnici. Chiar dacă le privim în contextul nivelului legăturii de date, ele sunt, de asemenea, aplicabile și altor niveluri.

3.5.1 Modele de tip automat finit

Un concept cheie folosit în multe modele de protocoale îl constituie **automatul finit**. Cu această tehnică, fiecare **automat al protocolului** (adică transmițător sau receptor) este în fiecare moment de timp într-o stare specifică. Stările sale constau din toate valorile variabilelor sale, incluzând contorul de instrucțiuni al programului.

În cele mai multe cazuri, un număr mare de stări pot fi grupate împreună, în vederea analizei. De exemplu, considerând receptorul din protocolul 3, am putea abstractiza toate stările posibile în două stări importante: așteptarea cadrului 0 sau așteptarea cadrului 1. Toate celelalte stări pot fi considerate ca fiind tranzitorii, simpli pași pe calea spre una din stările principale. De obicei, stările sunt alese ca fiind acele momente în care automatul protocolului așteaptă să se petreacă următorul eveniment [adică să execute apelul de procedură *wait(event)* din exemplele noastre]. În acest punct, starea automatului este complet determinată de stările variabilelor sale. Numărul de stări este deci 2^n, unde n este numărul de biți necesari pentru reprezentarea tuturor combinațiilor de variabile.

Starea întregului sistem este combinația tuturor stărilor celor două automate ale protocolului și a stării canalului. Starea canalului este determinată de conținutul său. Folosind din nou protocolul 3 ca exemplu, canalul are patru stări posibile: un cadru zero sau un cadru unu circulând de la transmițător la receptor, un cadru de confirmare circulând în sens invers sau nici un cadru. Dacă modelăm transmițătorul sau receptorul prin două stări, întregul sistem are 16 stări distincte.

Aici trebuie să spunem câteva cuvinte despre starea canalului. Conceptul de cadru circulând „prin canal" este, bineînțeles, o abstractizare. Adevăratul înțeles este acela că este posibil ca respectivul cadru să fi fost primit, dar nu și prelucrat la destinație. Un cadru rămâne „pe canal" până când automatul execută *FromPhysicalLayer* și îl prelucrează.

Din fiecare stare, există zero sau mai multe **tranziții** posibile spre alte stări. Tranzițiile au loc atunci când se petrece un eveniment. Pentru un automat, o tranziție trebuie să se facă atunci când este trimis un cadru, când sosește un cadru, când expiră un interval de timp, când apare o întrerupere etc. Pentru canal, evenimentele tipice sunt introducerea unui nou cadru pe canal de către automatul protocolului, livrarea cadrului unui automat sau pierderea unui cadru datorată unei rafale de zgomote. Dată fiind o descriere completă a automatelor protocolului și a caracteristicilor canalului, este posibil să trasăm graful orientat care prezintă toate stările automatului ca noduri și toate tranzițiile ca arce orientate.

O singură stare este desemnată ca **stare inițială**. Această stare corespunde descrierii sistemului, atunci când el începe să funcționeze, sau unui punct de pornire convenabil imediat următor. Unele stări, poate chiar toate stările, pot fi atinse din starea inițială printr-o secvență de tranziții. Folosind tehnicile binecunoscute din teoria grafurilor (de exemplu, calculul închiderii tranzitive a unui graf), este posibil să se determine care stări sunt accesibile și care nu. Această tehnică este numită **analiza accesibilității** (Lin ș.a., 1987). Această analiză poate fi utilă în determinarea corectitudinii protocolului.

Formal, un model de tip automat finit al unui protocol poate fi privit ca un cvadruplu *(S, M, I, T)* unde:

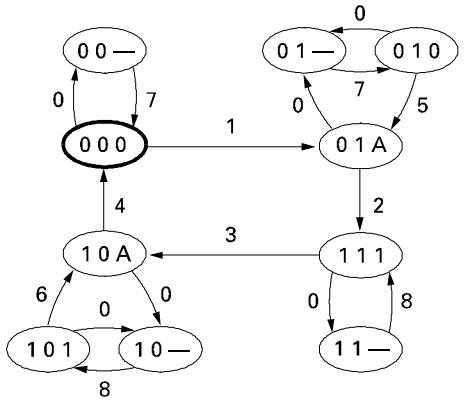
S este mulțimea stărilor în care se pot găsi procesele și canalul
M este mulțimea cadrelor care pot fi schimbate prin canal
I este mulțimea stărilor inițiale ale proceselor
T este mulțimea tranzițiilor între stări

La începutul intervalului de timp, toate procesele se găsesc în stările lor inițiale. Apoi încep să se producă evenimente, cum ar fi disponibilizarea unor cadre pentru transmisie sau expirarea unor intervale de timp. Fiecare eveniment poate face ca unul dintre procese sau canalul să execute o acțiune și să comute într-o nouă stare. Prin enumerarea atentă a fiecărui succesor posibil pentru fiecare stare, se poate construi graful de accesibilitate și se poate analiza protocolul.

Analiza accesibilității poate fi folosită pentru a detecta diferite erori în specificația protocolului. De exemplu, dacă este posibil ca un anumit cadru să apară într-o anumită stare și automatul finit să nu știe ce acțiune trebuie întreprinsă, atunci specificația este eronată (incompletitudine). Dacă există o mulțime de stări fără ieșire și din care nu se poate progresa, avem o altă eroare (interblocare). O eroare mai puțin serioasă este cea în care specificația protocolului spune cum să se trateze un eveniment într-o stare în care evenimentul nu se poate produce (tranziție neesențială). De asemenea pot fi detectate și alte erori.

Ca exemplu de model de automat finit să considerăm fig. 3-21(a). Acest graf corespunde protocolului 3 descris anterior: fiecare automat de protocol are două stări, iar canalul are patru stări. Există un total de 16 stări, nu toate accesibile din starea inițială. Stările inaccesibile nu sunt reprezentate

în figură. Fiecare stare este etichetată cu trei caractere, *SRC*, unde *S* este 0 sau 1, corespunzător cadrului pe care transmițătorul (S) încearcă să îl expedieze; *R* este de asemenea 0 sau 1, corespunzător cadrului pe care receptorul (R) îl așteaptă, iar *C* este 0, 1, *A* sau vid (-), corespunzător stării canalului. În acest exemplu, starea inițială a fost aleasă ca fiind (000). Cu alte cuvinte, transmițătorul tocmai a trimis cadrul 0, receptorul așteaptă cadrul 0 și cadrul 0 este actualmente pe canal.



Tranziție	Cine rulează?	Cadru acceptat	Cadru emis	Spre nivelul rețea
0	—	(cadru pierdut)		—
1	R	0	A	Da
2	S	A	1	—
3	R	1	A	Da
4	S	A	0	—
5	R	0	A	Nu
6	R	1	A	Nu
7	S	(timp expirat)	0	—
8	S	(timp expirat)	1	—

(b)

Fig. 3-21. (a) Diagrama de stare pentru protocolul 3. (b) Tranzițiile.

În fig. 3-21 sunt prezentate nouă tipuri de tranziții. Tranziția 0 corespunde pierderii conținutului canalului. Tranziția 1 corespunde livrării corecte a pachetului 0 la receptor, receptorul schimbându-și starea pentru a aștepta cadrul 1 și emițând o confirmare. Tranziția 1 include și livrarea pachetului 0 de către receptor spre nivelul rețea. Celelalte tranziții sunt listate în fig. 3-21(b). Sosirea unui cadru cu suma de control eronată nu a fost pusă în evidență, deoarece nu trebuie schimbată starea (în protocolul 3).

Pe parcursul operării normale, tranzițiile 1, 2, 3 și 4 sunt repetate în ordine, la nesfârșit. În fiecare ciclu sunt livrate două pachete, aducând transmițătorul înapoi în starea inițială, în care se încearcă transmiterea unui nou cadru cu numărul de secvență 0. În cazul în care canalul pierde cadrul 0, el face o tranziție din starea (000) în starea (00-). Până la urmă, transmițătorului îi expiră intervalul de timp (tranziția 7) și sistemul revine în starea (000). Pierderea unei confirmări este mai complicată, necesitând două tranziții, 7 și 5, sau 8 și 6, pentru a repara eroarea.

Una din proprietățile pe care protocolul cu număr de secvență pe 1 bit trebuie să le aibă este aceea că, indiferent de secvența de evenimente ce are loc, receptorul nu trebuie să livreze niciodată două pachete impare fără un pachet par intermediar și invers. Din graful din fig. 3-21 se vede că această cerință poate fi formulată mai riguros astfel: „nu trebuie să existe căi din starea inițială care să conțină două apariții ale tranziției 1 fără ca între ele să apară tranziția 3 sau invers.” Din figură se poate vedea că protocolul este corect în raport cu această cerință.

O cerință similară este aceea că nu trebuie să existe căi pe care transmițătorul să-și schimbe starea de două ori (de exemplu din 0 în 1 și înapoi în 0) în timp ce starea receptorului rămâne constantă. Dacă ar exista o astfel de cale, atunci, în secvența corespunzătoare de evenimente, două cadre ar fi iremediabil pierdute, fără ca receptorul să observe. Secvența de pachete livrată ar avea în ea o pierdere nedetectată a două pachete.

O altă proprietate importantă a unui protocol este absența interblocărilor. O **interblocare** (eng.: **deadlock**) este situația în care protocolul nu mai înregistrează nici un progres la transmitere (adică livrare de pachete spre nivelul rețea), indiferent de secvența de evenimente produse. În termenii modelului de graf, o interblocare este caracterizată de existența unei submulțimi de stări care este accesibilă din starea inițială și care are două proprietăți:

1. Nu există nici o tranziție într-o stare din afara submulțimii de stări.
2. În submulțimea de stări, nu există tranziții care să determine continuarea transmiterii.

Odată ajuns în situația de interblocare, protocolul rămâne aici pentru totdeauna. Din nou, este ușor de văzut din graf că protocolul 3 nu are interblocări.

3.5.2 Modele de tip rețea Petri

Automatul finit nu este singura tehnică de specificare formală a protocoalelor. În această secțiune vom descrie o altă tehnică, **Rețelele Petri** (Danthine, 1980). O rețea Petri are patru elemente de bază: locuri, tranziții, arce și jetoane. Un **loc** reprezintă o stare în care se poate găsi sistemul (sau o parte a sa). Fig. 3-22 prezintă o rețea Petri cu două locuri, *A* și *B*, reprezentate prin cercuri. Sistemul se află în starea *A*, indicată prin **jeton** (punctul îngroșat) în locul A. O **tranziție** este indicată printr-o bară orizontală sau verticală. Fiecare tranziție are zero sau mai multe **arce de intrare**, venind dinspre locuri de intrare, și zero sau mai multe **arce de ieșire**, mergând spre locuri de ieșire.

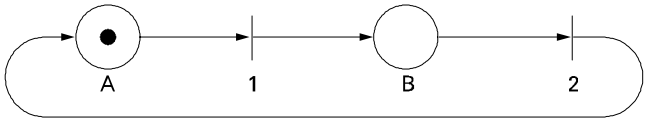


Fig. 3-22. O rețea Petri cu două locuri și două tranziții.

O tranziție este **activată** dacă există cel puțin un jeton de intrare în fiecare din locurile sale de intrare. Orice tranziție activată poate fi **executată** la dorință, ștergând un jeton din fiecare loc de intrare și depunând un jeton în fiecare loc de ieșire. Dacă numărul de arce de intrare și de ieșire diferă, jetoanele nu vor fi conservate. Dacă două sau mai multe tranziții sunt activate, oricare din ele se poate executa. Alegerea tranziției care se va executa este nedeterministă, motiv pentru care rețelele Petri sunt utile în modelarea protocoalelor. Rețeaua Petri din fig. 3-22 este determinată și poate fi folosită pentru a modela orice proces în două faze (de exemplu comportamentul unui bebeluș: mănâncă, doarme, mănâncă, doarme ș.a.m.d.). Ca în cazul tuturor instrumentelor de modelare, detaliile inutile sunt eliminate.

Fig. 3-23 dă modelul de tip rețea Petri pentru fig. 3-12. Spre deosebire de modelul de tip automat finit, aici nu există stări compuse: starea transmițătorului, starea canalului și starea receptorului sunt reprezentate separat. Tranzițiile 1 și 2 corespund trimiterii cadrului 0 de către transmițător, normal și, respectiv, la expirarea timpului. Tranzițiile 3 și 4 sunt analoagele pentru cadrul 1. Tranzițiile 5, 6 și 7 corespund pierderii unui cadru 0, unei confirmări și, respectiv, a unui cadru 1. Tranzițiile 8 și 9 se petrec atunci când la receptor sosește un cadru de date cu număr de secvență greșit. Tranzițiile 10 și 11 reprezintă sosirea la receptor a următorului cadrului din secvență și livrarea acestuia către nivelul rețea.

Rețelele Petri pot fi reprezentate într-o formă algebrică convenabilă asemănătoare gramaticilor. Fiecărei tranziții îi corespunde o regulă din gramatică. Fiecare regulă specifică locurile de intrare și de ieșire ale tranziției. Din moment ce fig. 3-23 are 11 tranziții, gramatica sa are 11 reguli, numerotate 1-11, fiecare corespunzând tranziției cu același număr. Gramatica pentru rețeaua Petri din fig. 3-23 este următoarea:

1: BD → AC
2: A → A
3: AD → BE
4: B → B
5: C →
6: D →
7: E →
8: CF → DF
9: EG → DG
10: CG → DF
11: EF → DG

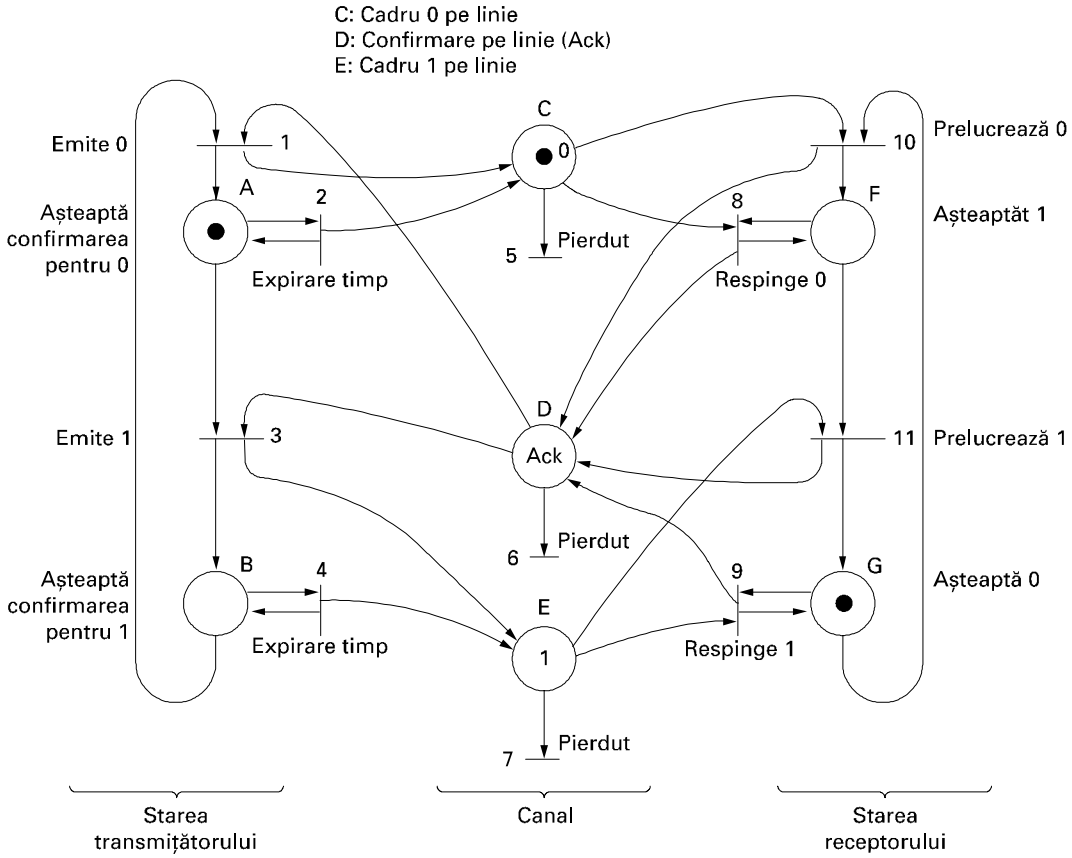


Fig. 3-23. Model de tip rețea Petri pentru protocolul 3.

Rețelele Petri pot fi utilizate pentru a detecta erori în protocol, într-un mod similar folosirii automatelor finite. De exemplu, dacă o secvență executabilă a inclus tranziția 10 de două ori fără a include tranziția 11 ca intermediar, protocolul ar fi incorect. Conceputul de interblocare într-o rețea Petri este de asemenea similar corespondentului său de la automatul finit.

Interesant de observat este cum s-a reușit reducerea unui protocol complex la 11 reguli gramaticale simple, care pot fi ușor manipulate de un program.

Starea curentă a rețelei Petri este reprezentată ca o colecție neordonată de locuri, fiecare loc fiind reprezentat în colecție de un număr de ori egal cu numărul de jetoane pe care le conține. Orice regulă ale cărei locuri din membrul stâng sunt prezente, poate fi executată, ștergând aceste locuri din starea curentă și adăugând locurile sale de ieșire la starea curentă. Marcajul din fig. 3-23 este ACG (adică A, C și G au fiecare câte un jeton), și astfel, regula 10 ($CG \rightarrow DF$) poate fi aplicată, ducând la o nouă stare (posibil cu același marcaj ca cea inițială), dar regula 3 ($AD \rightarrow BE$) nu poate fi aplicată.

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE

În următoarele secțiuni vom examina câteva protocoale larg folosite pentru legătura de date. Primul dintre ele, HDLC, este un protocol clasic orientat pe bit ale cărui variante sunt folosite de decenii întregi în multe aplicații. Cel de-al doilea, PPP, este protocolul de nivel legătură de date folosite pentru conectarea calculatoarelor casnice la Internet.

3.6.1 HDLC - Controlul de nivel înalt al legăturii de date

În această secțiune vom examina un grup de protocoale strâns legate, puțin mai vechi, dar care sunt încă foarte utilizate. Ele sunt toate derivate din protocolul pentru legătura de date utilizat în lumea mainframe-urilor IBM, numit SDLC (**Synchronous Data Link Control**, rom.: protocolul de control sincron al legăturii de date). După ce a dezvoltat SDLC, IBM l-a supus examinării ANSI și ISO pentru acceptare ca standard SUA și, respectiv, internațional. ANSI a modificat protocolul, astfel încât acesta a devenit **ADCCP** (**Advanced Data Communication Control Procedure**, rom.: procedură de control avansat al comunicațiilor de date), iar ISO l-a modificat și a produs **HDLC** (**High-level Data Link Control**, rom.: control de nivel înalt al legăturii de date). CCITT a adoptat și modificat HDLC pentru al său **LAP** (**Link Access Procedure**, rom.: procedură de acces la legătură) care este parte a standardului pentru interfața de rețea X.25, dar, mai târziu l-a modificat din nou, rezultând **LAPB**, în scopul de a-l face mai compatibil cu o versiune ulterioară de HDLC. Un lucru frumos în ceea ce privește standardele este că sunt multe, dintre care poți alege. În plus, dacă nu îți place nici unul dintre ele, poți aștepta modelul care va apărea anul viitor.

Aceste protocoale se bazează pe aceleași principii. Toate sunt orientate pe biți și folosesc inserarea de biți pentru transparența datelor. Ele diferă doar în puncte minore, și totuși supărătoare. Discuția care urmează, despre protocoalele orientate pe biți, intenționează a fi o introducere generală. Pentru detaliile specifice fiecărui protocol, consultați definiția corespunzătoare.

Toate protocoalele orientate pe biți folosesc structura de cadru prezentată în fig. 3-24. Câmpul *Adresă* este primul ca importanță pentru liniile cu terminale multiple, unde el este folosit pentru a

identifica unul dintre terminale. Pentru liniile punct-la-punct, el este folosit uneori pentru a deosebi comenzile de răspunsuri.

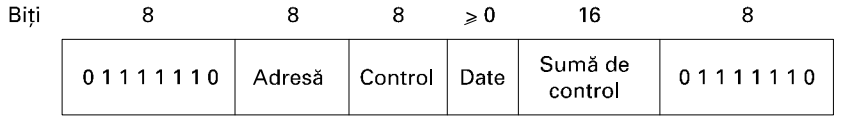


Fig. 3-24. Format de cadru pentru protocoalele orientate pe biți.

Câmpul *Control* este folosit pentru numere de secvență, confirmări și alte scopuri, după cum se va arăta în continuare.

Câmpul *Date* poate conține informații arbitrare. Poate avea lungime arbitrară, cu toate că eficiența sumei de control scade odată cu creșterea lungimii cadrului, datorită creșterii probabilității de apariție a erorilor în rafală.

Câmpul *Suma de Control* este o variantă CRC (Cyclic Redundancy Code - cod ciclic redundant), folosind tehnica prezentată in Sec. 3-2.2.

Cadrul este delimitat cu o altă secvență indicator (01111110). Pe liniile punct-la-punct inactive secvențele indicator sunt transmise continuu. Un cadru minim conține trei câmpuri și are în total 32 de biți, excluzând indicatorii de la capete.

Există trei tipuri de cadre: **Informație**, **Supervizor** și **Nenumerotat**. Conținutul câmpului *Control* pentru fiecare dintre aceste trei tipuri este prezentat în fig. 3-25. Acest protocol folosește o fereastră glisantă, cu un număr de secvență reprezentat pe 3 biți. În fereastră pot fi păstrate, la un moment dat, până la șapte cadre neconfirmate. Câmpul *Secvență* din fig. 3-25(a) este numărul de secvență al cadrului. Câmpul *Următor* este o confirmare atașată. Oricum, toate protocoalele aderă la convenția că, în loc să atașeze numărul ultimului cadru recepționat corect, să folosească numărul primului cadru nerecepționat (adică următorul cadru așteptat). Opțiunea pentru ultimul cadru primit sau următorul cadru recepționat este arbitrară; nu are importanță ce convenție este utilizată, dacă este folosită cu consecvență.

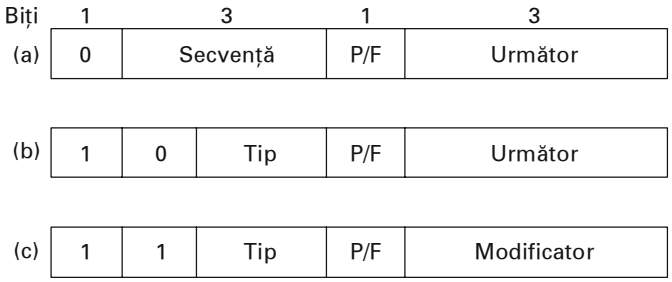


Fig. 3-25. Câmpul Control pentru (a) un cadru de informație, (b) un cadru de supervizare, (c) un cadru nenumerotat.

Bitul *P/F* înseamnă *Test/Final (eng.: Poll/Final)*. El este folosit atunci când un calculator (sau un concentrator) interoghează un grup de terminale. Când este folosit ca *P*, calculatorul invită terminalul să trimită date. Toate cadrele trimise de terminal, cu excepția celui final, au bitul *P/F* setat pe **P**. Pentru cadrul final bitul este setat la *F*.

În câteva dintre protocoale, bitul *P/F* este folosit pentru a forța cealaltă mașină să trimită imediat un cadru Supervizor, în loc să aștepte fluxul invers la care să se atașeze informația despre fereastră. Bitul are de asemenea câteva utilizări minore referitoare la cadrele nenumerotate.

Numeroasele tipuri de cadre Supervizor sunt diferențiate prin câmpul *Tip*. Tipul 0 este un cadru de confirmare (numit oficial RECEIVE READY) folosit pentru a indica următorul cadru așteptat. Cadrul este folosit atunci când nu există flux invers care să poată fi folosit pentru atașare.

Tipul 1 este un cadru de confirmare negativă (oficial numit REJECT). Este folosit pentru a indica detecția unei erori de transmisie. Câmpul *Următor* indică primul cadru din secvență ce nu a fost recepționat corect (deci cadrul ce trebuie retransmis). Transmițătorului i se cere să retransmită toate cadrele neconfirmate, începând cu *Următor*-ul. Această strategie este similară mai degrabă protocolului 5 decât protocolului 6.

Tipul 2 este RECEIVE NOT READY. El confirmă toate cadrele, cu excepția lui *Următor*, exact ca RECEIVE READY, dar spune transmițătorului să oprească transmisia. RECEIVE NOT READY este destinat să semnaleze anumite probleme temporare apărute la receptor, cum ar fi lipsa zonelor tampon, și nu ca o alternativă la controlul fluxului cu fereastră glisantă. Când problema a fost rezolvată, receptorul trimite un RECEIVE READY, REJECT sau anumite cadre de control.

Tipul 3 este SELECTIVE REJECT. El cere retransmiterea, însă doar pentru cadrul specificat. Din acest punct de vedere este mai apropiat de protocolul 6 decât de protocolul 5 și de aceea este folositor atunci când dimensiunea ferestrei transmițătorului este jumătate sau mai puțin din dimensiunea spațiului secvenței. Astfel, dacă receptorul dorește să păstreze cadre care erau în afara secvenței pentru posibila folosire ulterioară, el poate să forțeze retransmiterea oricărui cadru, folosind SELECTIVE REJECT. HDLC și ADCCP permit acest tip de cadru, dar SDLC și LAPB nu îl permit (adică nu există Selective Reject) și cadrele de tipul 3 nu sunt definite.

Cea de-a treia clasă o reprezintă cadrul Nenumerotat. El este folosit uneori în scopuri de control, dar poate fi folosit și pentru transportul datelor atunci când se recurge la un serviciu nesigur, neorientat pe conexiune. Diversele tipuri de protocoale orientate pe biți diferă considerabil aici, spre deosebire de celelalte două tipuri, unde erau aproape identice. Pentru a indica tipul cadrului sunt disponibili cinci biți, dar nu sunt folosite toate cele 32 de posibilități.

Toate protocoalele furnizează o comandă, DISC (DISConnect), care permite ca o mașină să anunțe că se va opri (de exemplu pentru întreținere preventivă). De asemenea există o comandă ce permite ca o mașină, care tocmai s-a reconectat, să-și anunțe prezența și să forțeze resetarea tuturor numerelor de secvență la zero. Această comandă poartă numele de SNRM (Set Normal Response Mode - stabilește modul normal de răspuns). Din nefericire, „modul normal de răspuns" numai normal nu este. Este un mod neechilibrat (adică asimetric) în care unul din capetele liniei este master iar celălalt este slave. SNRM datează din timpurile când comunicația datelor presupunea un terminal neinteligent comunicând cu un calculator gazdă puternic, ceea ce este, evident, asimetric. Pentru a face protocolul mai potrivit cazurilor în care cei doi parteneri sunt egali, HDLC și LAPB au o comandă suplimentară, SABM (Set Asynchronous Balanced Mode - stabilește modul asincron echilibrat), care resetează linia și declară ambii parteneri ca fiind egali. De asemenea, aceste protocoale au comenzile SABME și SNRME, care sunt identice cu SABM și, respectiv, SNRM, cu excepția faptului că ele permit folosirea unui format extins pentru cadru, care utilizează numere de secvență pe 7 biți în locul unora pe 3 biți.

A treia comandă prevăzută de toate protocoalele este FRMR (FRaMe Reject), folosită pentru a indica sosirea unui cadru cu suma de control corectă, dar cu semantică imposibilă. Exemple de semantică imposibilă sunt cadru de tipul 3 Supervizor în LAPB, un cadru mai scurt de 32 de biți, un

cadru de control nepermis, confirmarea unui cadru care a fost în afara ferestrei etc. Cadrele FRMR conțin un câmp de date de 24 de biți care arată ceea ce a fost eronat la cadrul respectiv. Datele includ câmpul de control al cadrului eronat, parametrii ferestrei și o colecție de biți folosiți pentru a semnala erori specifice.

Cadrele de control pot fi pierdute sau deteriorate ca și cadrele de date, de aceea și ele trebuie confirmate. În acest scop este furnizat un cadru special de control, numit UA (Unnumbered Acknowledgement). Deoarece poate exista un singur cadru de control neconfirmat, nu există niciodată ambiguități asupra cadrului care este confirmat.

Cadrele de control rămase sunt folosite pentru inițializare, interogare și raportarea stării. Există, de asemenea, un cadru de control care poate conține informații arbitrare, UI (Unnumbered Information). Aceste date nu sunt livrate nivelului rețea, ci sunt destinate a fi primite chiar de nivelul legătură de date.

În ciuda utilizării pe scară largă, HDLC este departe de a fi perfect. O discuție despre diversitatea problemelor asociate cu acest protocol poate fi găsită în (Fiorini ș.a., 1995).

3.6.2 Nivelul legăturii de date în Internet

Internet-ul constă din mașini individuale (calculatoare gazdă și rutere) și o infrastructură de comunicație care le conectează. În cadrul unei singure clădiri sunt larg utilizate LAN-urile pentru interconectare, dar infrastructura de arie largă este construită din linii închiriate, punct-la-punct. În Cap. 4 vom studia LAN-urile; aici vom examina protocoalele legăturii de date folosite pe liniile punct-la-punct în Internet.

În practică, comunicația punct-la-punct este folosită în principal în două situații. În primul rând, mii de organizații au una sau mai multe LAN-uri, fiecare cu un anumit număr de calculatoare gazdă (calculatoare personale, stații de lucru ale utilizatorilor, servere ș.a.m.d.) și un ruter (sau o punte care este funcțional similară). Adeseori, ruterele sunt interconectate printr-un trunchi LAN. În mod tipic, toate conexiunile cu lumea exterioară se fac printr-unul sau două rutere care au linii punct-la-punct închiriate spre rutere aflate la distanță. Internet-ul este construit din aceste rutere și liniile lor închiriate care realizează subrețelele de comunicație.

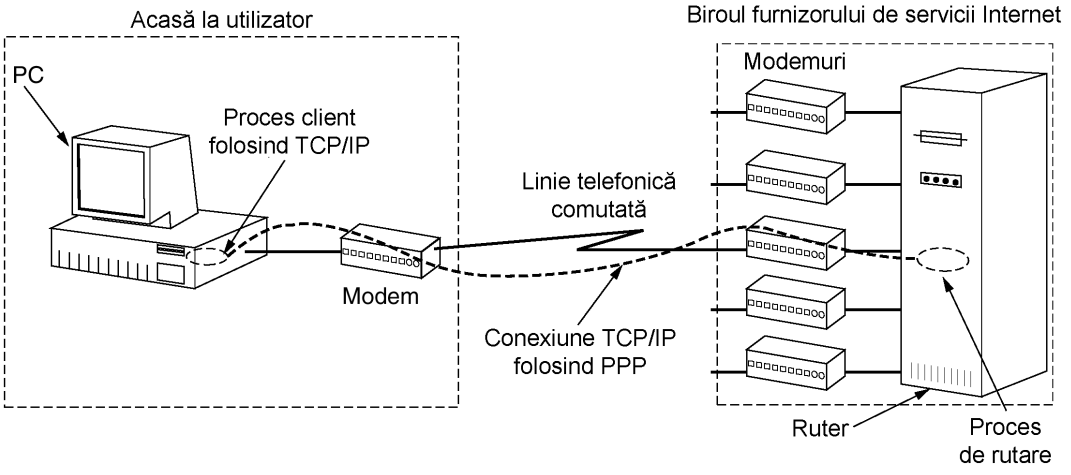


Fig. 3-26. Un calculator personal lucrând ca un calculator gazda în Internet.

A doua situație în care liniile punct-la-punct joacă un rol major în Internet o reprezintă milioanele de utilizatori individuali care au conexiuni de acasă la Internet folosind modemuri și linii telefonice comutate. De obicei PC-ul de acasă al unui utilizator apelează ruterul unui **furnizor de servicii Internet**, și poate acționa astfel ca orice calculator gazdă Internet. Această metodă de operare nu este diferită de aceea în care există o linie închiriată între PC și ruter, cu excepția faptului că atunci când utilizatorul își termină sesiunea, conexiunea va fi închisă. În fig. 3-26 este ilustrat un PC casnic ce apelează un furnizor de servicii Internet. Modem-ul este prezentat ca fiind extern, tocmai pentru a-i accentua rolul, dar calculatoarele moderne dispun de modem-uri interne.

Atât pentru conexiunea pe linie închiriată ruter-ruter cât și pentru conexiunea comutată calculator gazdă-ruter, este necesar un protocol de legătură de date punct-la-punct pentru încadrare, controlul erorilor și pentru alte funcții ale nivelului legătură de date pe care le-am studiat în acest capitol. Cel folosit în Internet este PPP. În continuare acesta va fi prezentat în detaliu.

PPP - Point-to-Point Protocol (rom.: protocol punct-la-punct)

Internetul are nevoie de un protocol punct-la-punct care să servească mai multor scopuri, includzând trafic ruter-la-ruter și trafic utilizator-la-ISP. Acest protocol este **PPP** (**Point-to-Point Protocol**, rom.: **protocolul punct-la-punct**), care este definit in RFC 1661 și dezvoltat în alte câteva RFC-uri (de exemplu RFC-urile 1662, 1663). PPP face detecția erorilor, suportă mai multe protocoale, permite ca adresele IP să fie negociate în momentul conectării, permite autentificarea și are multe alte capabilități.

PPP furnizează trei lucruri:

1. O metodă de împărțire în cadre care delimitează, fără ambiguitate, sfârșitul unuia și începutul următorului. Formatul cadrului permite și detecția de erori.
2. Un protocol de control al legăturii pentru a obține liniile, a le testa, a negocia opțiunile și pentru a elibera liniile atunci când nu mai este nevoie de ele. Acest protocol se numește **LCP**(**Link Control Protocol**, rom: **protocolul de control al legăturii**). El suportă circuite sincrone și asincrone și codificări orientate atât pe bit, cât și pe caracter.
3. Un mod de a negocia opțiunile nivelului rețea într-un mod independent de protocolul folosit pentru nivelul rețea. Metoda aleasă este de a avea un **NCP** (**Network Control Protocol**, rom: **protocol de control al rețelei**) pentru fiecare nivel de rețea suportat.

Pentru a vedea cum lucrează împreună aceste părți, să considerăm un scenariu tipic în care un utilizator sună de la domiciliu un furnizor de servicii Internet pentru a transforma PC-ul său de acasă într-un calculator gazdă Internet temporar. PC-ul apelează mai întâi ruterul furnizorului prin intermediul unui modem. După ce modemul ruterului a răspuns la telefon și s-a stabilit o conexiune fizică, PC-ul trimite ruterului o serie de pachete LCP în câmpul de informație utilă (payload) al unuia sau mai multor cadre PPP. Aceste pachete și răspunsurile lor selectează parametrii PPP ce vor fi utilizați.

Odată ce parametrii s-au stabilit de comun acord, mai multe pachete NCP sunt trimise pentru a configura nivelul rețea. În mod obișnuit, PC-ul vrea să ruleze o suită de protocoale TCP/IP și va avea nevoie de o adresă IP. Deoarece nu există adrese IP suficiente, fiecare furnizor de Internet ia o parte din ele și asociază dinamic câte una pentru fiecare PC atașat în rețea, pe durata sesiunii de conectare. Dacă un furnizor posedă n adrese IP, el poate avea până la n mașini conectate simultan, dar numărul total de clienți poate fi de mai multe ori pe atât. NCP pentru IP este folosit pentru a realiza asocierea adreselor IP.

În acest moment, PC-ul este un calculator gazdă Internet și poate trimite și primi pachete IP, exact așa cum o pot face calculatoarele conectate prin cabluri. Când utilizatorul termină, NCP întrerupe conexiunea la nivelul rețea și eliberează adresa IP. Apoi LCP întrerupe conexiunea la nivelul legătură de date. În final, calculatorul spune modemului să închidă telefonul, eliberând conexiunea la nivel fizic.

Formatul cadrului PPP a fost ales foarte asemănător cu formatul cadrului HDLC deoarece nu exista nici un motiv pentru a se reinventa roata. Diferența majoră între PPP și HDLC este că primul este mai degrabă orientat pe caractere decât pe biți. În particular, PPP folosește umplerea cu caractere pe liniile comutate prin modem, astfel încât toate cadrele au un număr întreg de octeți. Nu este posibil să se trimită un cadru constând din 30.25 octeți, așa cum era la HDLC. Cadrele PPP pot fi transmise nu numai pe liniile telefonice comutate, ele pot fi transmise și pe linii SONET sau linii HDLC, cu adevărat orientate pe biți (de exemplu pentru conexiuni ruter-ruter). Formatul cadrului PPP este prezentat în fig. 3-27.

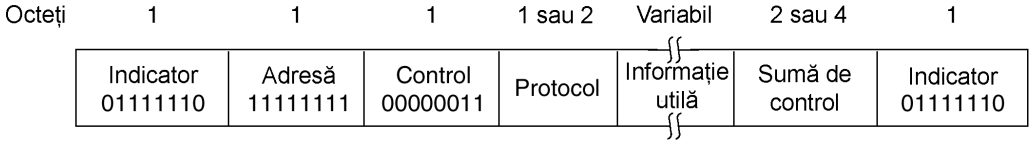


Fig. 3-27. Formatul complet de cadru PPP pentru operarea în mod nenumerotat.

Toate cadrele PPP încep cu octetul indicator HDLC standard (01111110), pentru care se folosește umplerea cu caractere, dacă apare în cadrul câmpului ce specifică informația utilă. După acesta urmează câmpul *Adresă*, care este întotdeauna setat la valoarea binară 11111111, indicând astfel că toate stațiile trebuie să accepte cadrul. Folosirea acestei valori evită problema necesității de a se asocia adrese legăturii de date.

Câmpul *Adresă* este urmat de câmpul *Control*, a cărui valoare implicită este 00000011. Această valoare indică un cadru nenumerotat. Cu alte cuvinte, PPP nu oferă o transmisie sigură folosind numere de secvență și confirmări în mod implicit. În medii cu zgomote, cum ar fi rețelele fără fir, poate fi folosită transmisia sigură utilizând numere de secvență. Detaliile exacte sunt definite în RFC 1663, dar această facilitate este rar utilizată.

Deoarece câmpurile *Adresă* și *Control* sunt întotdeauna constante în configurațiile implicite, LCP furnizează mecanismul necesar ca cele două părți să negocieze opțional omiterea amândurora și să economisească astfel doi octeți pe cadru.

Cel de-al patrulea câmp PPP este câmpul *Protocol*. Sarcina lui este să spună ce tip de pachet este în câmpul *Informație utilă*. Sunt definite coduri pentru LCP, NCP, IP, IPX, AppleTalk și alte protocoale. Protocoalele ce încep cu un bit 0 sunt protocoale pentru nivelul rețea. cum ar fi IP, IPX, OSI CLNP, XNS. Acelea care încep cu un bit 1 sunt folosite pentru a negocia alte protocoale. Acestea includ LCP și un NCP diferit pentru fiecare protocol de rețea suportat. Dimensiunea implicită a câmpului *Protocol* este de 2 octeți, dar ea poate fi negociată la 1 octet folosind LCP.

Câmpul *Informație utilă* este de lungime variabilă, până la o anumită limită maximă negociată. Dacă lungimea nu este negociată folosind LCP în timpul setării liniei, este folosită o lungime implicită de 1500 de octeți. Dacă este necesar, după informația utilă pot fi adăugate caractere de umplere.

După câmpul *Informație utilă* urmează câmpul S*umă de control*, care este în mod normal de 2 octeți, dar poate fi modificat la 4 octeți.

În concluzie, PPP este un mecanism de încadrare multiprotocol potrivit pentru folosirea pe linii cu modem, linii seriale orientate pe biți HDLC, SONET și alte niveluri fizice. Suportă detecția erorilor, negociere opțională, compresia antetului și, opțional, transmisie sigură folosind cadre HDLC.

Să ne întoarcem acum de la formatul cadrului PPP la modul în care liniile sunt stabilite (eng.: brought up) și eliberate (eng.: brought down). Diagrama simplificată din fig. 3-28 arată fazele prin care trece o linie atunci când este stabilită, folosită și eliberată. Secvența se aplică atât pentru conexiunile prin modem cât și pentru conexiunile ruter-ruter.

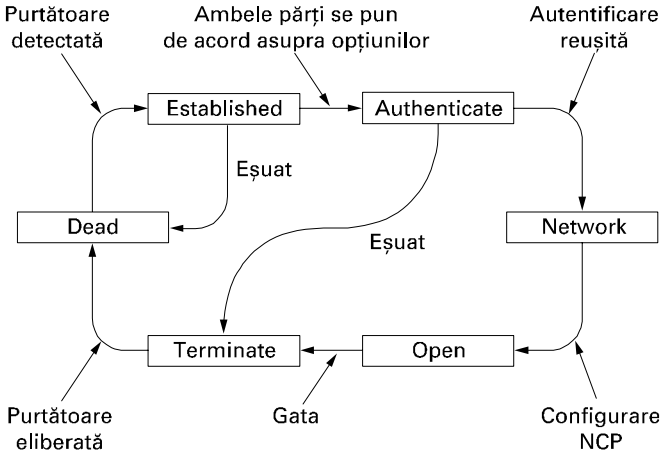


Fig. 3-28. O diagramă de faze simplificată pentru stabilirea și eliberarea unei linii.

Protocolul se inițializează cu linia în starea *DEAD*, stare care semnifică faptul că nu este prezentă nici o purtătoare la nivel fizic și nu există nici o conexiune fizică. După ce este stabilită conexiunea fizică, linia trece în *ESTABLISH*. În acest punct începe negocierea opțională LCP care, dacă reușește, conduce la *AUTHENTICATE*. Acum cele două părți pot să-și verifice una alteia identitatea, dacă doresc. Când se intră în faza *NETWORK*, este invocat protocolul NCP corespunzător pentru a configura nivelul rețea. Dacă configurarea se face cu succes, este atinsă faza *OPEN* și poate avea loc transportul datelor. Când transportul datelor este terminat, linia este trecută în faza *TERMINATE* și, de aici, înapoi în *DEAD* unde purtătoarea este întreruptă.

Nume	Direcție	Descriere
Configure-request	I→R	Lista opțiunilor și valorilor propuse
Configure-ack	I←R	Toate opțiunile sunt acceptate
Configure-nak	I←R	Anumite opțiuni nu sunt acceptate
Configure-reject	I←R	Anumite opțiuni nu sunt negociabile
Terminate-request	I→R	Cerere de eliberare a liniei
Terminate-ack	I←R	OK, linia este eliberată
Code-reject	I←R	Primire cerere necunoscută
Protocol-reject	I←R	Cerere protocol necunoscut
Echo-request	I→R	Rog trimiterea acestui cadru înapoi
Echo-replay	I←R	Iată cadrul înapoi
Discard-request	I→R	Ignoră cadrul (pentru testare)

Fig. 3-29. Tipurile de cadre LCP.

LCP negociază opțiunile protocolului legăturii de date în timpul fazei *ESTABLISH*. Protocolul LCP nu se ocupă chiar de opțiuni, ci de mecanismul de negociere. El furnizează procesului inițiator un mod de a face o propunere și procesului de răspuns un mod de a accepta sau refuza această propunere. De asemenea, el furnizează celor două procese un mecanism de a testa calitatea liniei, de a verifica dacă aceasta este suficient de bună pentru a defini o conexiune. În fine, protocolul LCP permite liniilor să fie eliberate atunci când nu mai este nevoie de ele.

În RFC 1661 sunt definite unsprezece tipuri de cadre LCP. Acestea sunt listate în fig. 3-29. Cele patru tipuri *Configure-* permit inițiatorului (I) să propună valori pentru opțiuni și celui care răspunde (R) să le accepte sau să le refuze. În ultimul caz, cel care răspunde poate face o propunere alternativă sau poate anunța că nu este gata să negocieze în nici un fel anumite opțiuni. Opțiunile ce vor fi negociate și valorile propuse pentru ele sunt conținute în cadrele LCP.

Codurile *Terminate-* sunt folosite pentru a elibera o linie atunci când ea nu mai este necesară. Codurile *Code-reject* și *Protocol-reject* sunt folosite de către cel ce răspunde pentru a spune că a primit ceva ce nu înțelege. Această situație poate însemna că a avut loc o eroare de transmisie, dar, mai degrabă, înseamnă că inițiatorul și cel ce răspunde folosesc versiuni diferite ale protocolului LCP. Tipurile *Echo-* sunt folosite pentru a testa calitatea liniei. În sfârșit, *Discard-request* este folosit pentru depanare. Dacă unul din capete are probleme cu transmiterea biților, programatorul poate folosi acest tip pentru testare. Dacă el reușește să meargă de la un capăt la celălalt, receptorul doar îl rejectează, fără a întreprinde nici o acțiune care ar putea genera confuzii pentru persoana care testează.

Opțiunile care pot fi negociate includ definirea dimensiunii maxime pentru informația utilă din cadrele de date, activarea autentificării și alegerea protocolului ce va fi folosit, activarea monitorizării calității liniei în timpul operațiunilor normale și selectarea diferitelor opțiuni pentru comprimarea antetului.

Nu se pot spune multe despre protocoalele NCP în general. Fiecare este specific unui anumit protocol de nivel rețea și permite să se facă cereri de configurare ce sunt specifice unui anumit protocol. De exemplu, pentru IP, asocierea dinamică a adreselor este cea mai importantă posibilitate.

3.7 REZUMAT

Sarcina nivelului legătură de date este de a converti șirurile de biți oferite de nivelul fizic în șiruri de cadre pentru a fi folosite de către nivelul rețea. Sunt utilizate diferite metode de încadrare, incluzând numărarea caracterelor, inserarea de octeți și umplerea cu biți. Protocoalele legăturii de date pot oferi controlul erorilor pentru retransmiterea cadrelor distruse sau pierdute. Pentru a împiedica un emițător rapid să suprasolicite un receptor lent, protocolul legăturii de date poate realiza și controlul fluxului. Mecanismul cu fereastră glisantă este foarte folosit pentru a integra controlul erorilor și controlul fluxului într-un mod convenabil.

Protocoalele cu fereastră glisantă pot fi clasificate după dimensiunea ferestrei emițătorului și după dimensiunea ferestrei receptorului. Când ambele sunt egale cu 1, protocolul este pas-cu-pas (eng.: stop-and-wait). Când fereastra emițătorului este mai mare ca 1, de exemplu pentru a împiedica emițătorul să blocheze un circuit cu o întârziere mare de propagare, receptorul poate fi programat fie să elimine toate celelalte cadre cu excepția următorului din secvență, fie să memoreze cadrele neordonate până când ele vor fi necesare.

În acest capitol au fost prezentate o serie de protocoale. Protocolul 1 a fost conceput pentru un mediu fără erori, în care receptorul poate face față oricărui flux de la emițător. Protocolul 2 presupune existența unui mediu fără erori, dar introduce controlul fluxului. Protocolul 3 tratează problema erorilor prin utilizarea numerelor de secvență și a algoritmului pas-cu-pas. Protocolul 4 permite comunicația bidirecțională și introduce conceptul de atașare (eng.: piggybacking). Protocolul 5 folosește un protocol cu fereastră glisantă și revenire cu n pași (eng.: go back n). Protocolul 6 folosește repetarea selectivă și confirmări negative.

Protocoalele pot fi modelate folosind diferite tehnici ce ajută la demonstrarea corectitudinii lor (sau a lipsei acesteia). Modelele bazate pe automate finite și modelele bazate pe rețele Petri sunt larg utilizate în acest scop.

Multe rețele folosesc la nivelul legătură de date unul dintre protocoalele orientate pe biți - SDLC, HDLC, ADCCP sau LAPB. Toate aceste protocoale folosesc octeți indicatori pentru delimitarea cadrelor și inserarea de biți pentru a preveni apariția octeților indicatori în cadrul datelor. De asemenea toate aceste protocoale folosesc fereastra glisantă pentru controlul fluxului. Internet-ul folosește PPP ca principal protocol al legăturii de date pe liniile de tip punct-la-punct.

3.8 PROBLEME

1. Un mesaj de la un nivel mai înalt este spart în 10 cadre, fiecare dintre acestea având 80% șansă de a ajunge nemodificat. Dacă nu se face nici un control al erorilor de către protocolul legăturii de date, de câte ori va trebui transmis mesajul în medie pentru a-l obține întreg la destinație?

2. Următoarea codificare a caracterelor este utilizată în cadrul unui protocol de nivel legătură de date:

A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000

Determinați secvența de biți transmisă (în binar) pentru cadrul format din următoarele 4 caractere: A B ESC FLAG, când fiecare din metodele de încadrare următoare sunt utilizate:
a) numărarea caracterelor
b) octeți indicatori și inserarea de octeți
c) octeți indicatori de început și sfârșit, cu inserare de biți.

3. Următorul fragment de date apare în mijlocul unui șir de date pentru care este folosit algoritmul de inserare de octeți descris în text: A B ESC C ESC FLAG FLAG D. Care este ieșirea după inserare?

4. Unul dintre colegii Dvs. de clasă, Scrooge, a remarcat faptul că este ineficientă folosirea a 2 octeți indicatori, unul pentru începutul cadrului, celălalt pentru sfârșit. Un singur octet indicator ar fi suficient, câștigându-se astfel un octet. Sunteți de acord?

5. Dacă în șirul de biți 0111101111101111110 se inserează biți, care este șirul de ieșire?

6. Când este utilizată inserarea de biți, este posibil ca prin pierderea, inserarea sau modificarea unui singur bit să se provoace o eroare nedetectabilă prin suma de control? Dacă nu, de ce? Dacă da, de ce? Lungimea sumei de control joacă vreun rol aici?

7. Puteți concepe o situație în care un protocol cu buclă deschisă (de exemplu un cod Hamming) poate fi preferabil protocoalelor cu buclă de reacție (feedback), discutate pe parcursul acestui capitol?

8. Pentru a oferi o siguranță mai mare decât cea pe care o poate da un singur bit de paritate, o schemă de codificare cu detecție de erori folosește un bit de paritate pentru verificarea tuturor biților de ordin impar și un al doilea bit de paritate pentru toți biții de ordin par. Care este distanța Hamming pentru un astfel de cod?

9. Se transmit mesaje de 16 biți folosind un cod Hamming. Câți biți de control sunt necesari pentru a asigura detectarea și corectarea de către receptor a erorilor de un bit? Prezentați secvența de biți transmisă pentru mesajul 1101001100110101. Presupuneți că se folosește paritare pară.

10. Un octet (8 biți) cu valoarea binară 10101111 trebuie codificat utilizând un cod Hamming cu paritate pară. Care este valoarea binară după codificare?

11. Un cod Hamming de 12 biți a cărui valoare în hexazecimal este 0xE4F sosește la receptor. Care este valoarea hexazecimală originală? Presupuneți ca maxim un bit este eronat.

12. Un mod de a detecta erorile este de a transmite datele ca un bloc de n rânduri a câte k biți pe rând și adăugarea de biți de paritate pentru fiecare rând și fiecare coloană. În colțul din dreapta jos este bitul de paritate care verifică linia și coloana sa. Va detecta această schemă toate erorile singulare? Dar erorile duble? Dar erorile triple?

13. Un bloc de biți cu *n* rânduri și *k* coloane folosește biți de paritate verticală și orizontală pentru detecția erorilor, Să presupunem că datorită erorilor de transmisie sunt inversați exact 4 biți. Deduceți o expresie pentru exprimarea probabilității ca eroarea să nu fie detectată.

14. Ce rest se obține prin împărțirea lui x^7+x^5+1 la polinomul generator x^3+1?

15. Secvența de biți 10011101 este transmisă folosind metoda CRC descrisă anterior. Polinomul generator este x^3+1. Prezentați secvența de biți transmisă. Se presupune că al treilea de la stânga este inversat în timpul transmisiei. Arătați că această eroare este detectată de receptor.

16. Protocoalele legăturii de date pun aproape întotdeauna CRC-ul în partea finală și nu în antet. De ce?

17. Un canal are o rată de transmisie a biților de 4 Kbps și o întârziere de propagare de 20 ms. Pentru ce domeniu al dimensiunii cadrelor metoda pas-cu-pas (stop-and-wait) are o eficiență de cel puțin 50%?

18. Un trunchi T1 lung de 3000 km este folosit pentru a transmite cadre de 64 de biți folosind protocolul 5. Dacă viteza de propagare este de 6 μsec/km, pe câți biți trebuie reprezentate numerele de secvență?

19. În protocolul 3, este posibil ca emițătorul să pornească contorul de timp, atunci când acesta merge deja? Dacă da, când se poate întâmpla acest lucru? Daca nu, de ce este imposibil?

20. Imaginați un protocol cu fereastră glisantă ce folosește suficienți biți pentru numerele de secvență, astfel încât să nu apară niciodată suprapuneri. Ce relație trebuie să existe între cele patru limite ale ferestrelor și dimensiunea ferestrei?

21. Dacă în procedura *between* din protocolul 5 este verificată condiția $a \leq b \leq c$ în locul condiției $a \leq b < c$, ar avea aceasta vreun efect asupra corectitudinii protocolului sau eficienței sale? Explicați răspunsul.

22. În protocolul 6, când sosește un cadru de date, este făcută o verificare pentru a se vedea dacă numărul de secvență diferă de cel așteptat și *no_nak* este adevărat. Dacă ambele condiții sunt îndeplinite, este trimis un NAK. Altfel, este pornit contorul de timp auxiliar. Presupuneți că ar fi omisă clauza else. Ar afecta aceasta corectitudinea protocolului?

23. Presupunem că bucla while cu trei instrucțiuni din finalul protocolului 6 a fost ștearsă din cod. Ar afecta aceasta corectitudinea protocolului sau doar performanța? Explicați răspunsul.

24. Presupunem că instrucțiunea *case* pentru erorile de sumă de control a fost scoasă din instrucțiunea *switch* din protocolul 6. Cum ar afecta aceasta operarea protocolului?

25. În protocolul 6 codul pentru *frame_arrival* are o secțiune folosită pentru NAK-uri. Această secțiune este invocată în cazul în care cadrul sosit este un NAK și este îndeplinită încă o condiție. Indicați un scenariu în care prezența acestei condiții este esențială.

26. Imaginați-vă că scrieți un program la nivelul legătură de date pentru o linie folosită pentru a primi date, dar nu și pentru a trimite. Celălalt capăt folosește HDLC, cu un număr de secvență pe 3 biți și o dimensiune a ferestrei de 7 cadre. Ați dori să memorați cât mai multe cadre din secvență pentru a crește eficiența, dar nu vă este permis să modificați programul transmițătorului. Este posibil să aveți o fereastră la receptor mai mare ca 1 și totuși să existe garanția că protocolul nu va eșua? Dacă da, care este fereastra cea mai mare care poate fi utilizată în siguranță?

27. Considerați operarea protocolului 6 pe o linie fără erori de 1 Mbps. Dimensiunea maximă a cadrului este 1000 biți. Pachetele noi sunt generate la un interval de aproape o secundă. Intervalul de expirare a timpului este de 10 ms. Dacă ar fi eliminate confirmările speciale pentru contorul de timp, ar putea apărea expirări de timp inutile. De câte ori ar trebui transmis în medie un mesaj?

28. În protocolul 6, $MAX\_SEQ = 2^n - 1$. Deși această condiție este evident necesară pentru a utiliza eficient biții din antet, nu s-a demonstrat că ea este și esențială. Ar funcționa protocolul corect pentru $MAX\_SEQ = 4$ de exemplu?

29. Cadrele de 1000 de biți sunt transmise pe un canal printr-un satelit geostaționar de 1 Mbps a cărui întârziere de propagare de la Pământ este de 270 milisecunde. Confirmările sunt întotdeauna atașate cadrelor de date. Antetele sunt foarte scurte. Sunt folosite numere de secvență pe 3 biți. Care este utilizarea maximă realizabilă a canalului pentru:

a) Pas-cu-pas (stop-and-wait);
b) Protocolul 5;
c) Protocolul 6.

30. Calculați fracțiunea din lărgimea de bandă ce este pierdută datorită supraîncărcării (antete și retransmisie) pentru protocolul 6 pe un canal de satelit de 50 Kbps, foarte încărcat cu cadre de date constând din 40 de biți antet și 3960 biți de date. Presupuneți o întârziere de la Pământ la satelit de 270 milisecunde. Cadrele ACK nu apar niciodată. Cadrele NAK sunt de 40 de biți. Rata de erori pentru cadrele de date este de 1% și rata de erori pentru cadrele NAK este neglijabilă. Numerele de secvență sunt pe 8 biți.

31. Se consideră un canal prin satelit fără erori, de 64 Kbps, folosit pentru a transmite cadre de date de 512 octeți într-o singură direcție, cu confirmări foarte scurte ce se întorc pe cealaltă cale. Care este productivitatea maximă pentru dimensiuni ale ferestrei de 1, 7, 15 și 127? Presupuneți o întârziere de la Pământ la satelit de 270 milisecunde.

32. Un cablu lung de 100 km funcționează la rata de transmisie de date T1. Viteza de propagare pe cablu este 2/3 din viteza luminii. Câți biți încap pe cablu?

33. Se presupune că se modelează protocolul 4 utilizând modelul automatelor finite. Câte stări există pentru fiecare mașină? Câte stări există pentru canalul de comunicație? Dar pentru un sistem complet (două mașini și canalul)? Se ignoră erorile de sumă de control.

34. Determinați o secvență executabilă pentru rețeaua Petri din fig. 3-23 corespunzătoare secvenței de stări (000), (01A), (01-), (010), (01A) în fig. 3-21. Explicați în cuvinte ce reprezintă secvența respectivă.

35. Date fiind regulile de tranziție AC→B, B→AC, CD→E și E→CD, desenați rețeaua Petri descrisă de ele. Folosind rețeaua Petri, desenați graful finit al stărilor accesibile din starea inițială ACD. Ce concept bine-cunoscut din știința calculatoarelor folosește acest model de reguli de tranziție?

36. PPP se bazează pe HDLC, care folosește inserarea de biți pentru a împiedica octeții indicatori accidentali din interiorul informației utile să provoace confuzii. Dați cel puțin un motiv pentru care PPP folosește în locul acesteia inserarea de octeți.

37. Care este supraîncărcarea minimă în transmiterea unui pachet IP folosind PPP? Luați în considerare doar supraîncărcarea introdusă de PPP însuși, nu și supraîncărcarea produsă de antetul IP.

38. Scopul acestui exercițiu este implementarea unui mecanism de detectare a erorilor folosind algoritmul standard CRC prezentat în text. Scrieți două programe, *generator* și *verificator*. Programul generator citește de la intrarea standard mesaje de n biți ca șiruri de 1 și 0, ca o linie de text ASCII. A doua linie este polinomul generator pe k biți, citit tot ca text ASCII. La ieșire, programul va afișa la ieșirea standard (standard output) o linie de text ASCII cu $n+k$ caractere 0 și 1, reprezentând mesajul de transmis. Apoi afișează polinomul, așa cum l-a citit. Programul verificator citește ieșirea programului generator și afișează un mesaj care indică dacă aceasta este corectă sau nu. Se va scrie apoi un program, *alterează*, care inversează un bit din prima linie depinzând de unul din parametri cu care a fost apelat (bitul cel mai din stânga se consideră bitul 1), dar copiază restul corect. Tastând:

generator < fișier | verificator

ar trebui să vedeți că mesajul este corect, dar tasând:

generator < fișier | alterează argument | verifica

ar trebui să primiți un mesaj de eroare.

39. Scrieți un program care să simuleze comportamentul unei rețele Petri. Programul trebuie să citească regulile de tranziție și o listă de stări corespunzând nivelului legătură al rețelei ce emite un nou pachet sau acceptă un pachet. Din starea inițială, de asemenea citită de pe mediul de intrare, programul trebuie să aleagă tranzițiile permise și să le execute aleator, verificând dacă un calculator gazdă acceptă două mesaje fără ca un alt calculator gazdă să emită unul nou între ele.

4

SUBNIVELUL DE ACCES LA MEDIU

Așa cum am arătat în Cap. 1, rețelele pot fi împărțite în două categorii: cele care utilizează conexiuni punct-la-punct și cele care utilizează canale cu difuzare (broadcast channels). Acest capitol se ocupă de rețelele cu difuzare (broadcast networks) și de protocoalele lor.

În orice rețea cu difuzare, una dintre probleme este determinarea utilizatorului cu drept de acces la canal în cazul în care există mai mulți utilizatori concurenți. Pentru a lămuri lucrurile, să considerăm o teleconferință în care șase persoane, vorbind de la șase telefoane diferite, sunt conectate astfel încât fiecare îi poate auzi pe ceilalți și poate vorbi cu ei. Este foarte probabil ca atunci când cineva se oprește din vorbit, doi sau mai mulți să înceapă să vorbească simultan, ceea ce va duce la haos. Într-o întâlnire față-în-față, haosul este evitat prin mijloace externe - de exemplu, prin ridicarea mâinii pentru a cere permisiunea de a vorbi. Când este disponibil un singur canal de comunicație, este mult mai greu să determini cine urmează să ia cuvântul. Sunt cunoscute multe protocoale de rezolvare a acestei probleme și ele constituie conținutul acestui capitol. În literatura de specialitate, canalele cu difuzare sunt uneori numite **canale multiacces** (multiaccess channels), sau **canale cu acces aleator** (random access channels).

Protocoalele folosite pentru a determina cine urmează într-un canal multiacces aparțin unui subnivel al nivelului legătură de date, numit subnivelul **MAC** (**Medium Access Control**, rom: controlul accesului la mediu). Subnivelul MAC este important mai ales pentru rețelele de tip LAN – Local Area Network (le vom numi prescurtat LAN-uri), care utilizează aproape toate un canal multiacces ca bază pentru comunicație. Din contră, rețelele de tip WAN – Wide Area Network (le vom numi WAN-uri) utilizează legături punct-la-punct, cu excepția rețelelor prin satelit. Datorită faptului că LAN-urile și canalele multiacces sunt atât de strâns legate, în acest capitol vom discuta la modul general atât despre LAN-uri cât și despre rețele prin satelit și alte rețele cu difuzare. Deoarece canalele multiaccess și rețelele locale sunt subiecte atât de apropiate, în acest capitol vom discuta mai

multe subiecte legate de rețelele locale în general, incluzând și subiecte care nu sunt strict specifice subnivelului MAC.

Tehnic vorbind, subnivelul MAC reprezintă partea de jos a nivelului legătură de date, deci logic ar fi fost să îl fi studiat înainte de a trece în revistă toate protocoalele punct-la-punct din cap. 3. Dar, pentru majoritatea oamenilor, înțelegerea protocoalelor care implică mai multe părți este mai ușoară după ce au înțeles bine protocoalele care implică numai două părți. Pentru acest motiv am deviat puțin de la stilul de prezentare strict ascendent al ierarhiei rețelelor.

4.1 PROBLEMA ALOCĂRII CANALULUI

Tema centrală a acestui capitol o reprezintă modul de alocare a unui singur canal cu difuzare între mai mulți utilizatori concurenți. Mai întâi vom arunca o privire de ansamblu asupra schemelor statice și dinamice de alocare. Apoi vom studia câțiva algoritmi specifici.

4.1.1 Alocarea statică a canalului în rețelele LAN și MAN

Modul tradițional de alocare a unui singur canal, cum ar fi cablul telefonic, între mai multi utilizatori concurenți este multiplexarea cu diviziunea frecvenței (FDM – Frequency Division Multiplexing). Dacă există N utilizatori, banda de transmisie este împărțită în N părți egale (vezi fig. 2-24), fiecărui utilizator fiindu-i alocată una dintre acestea. Deoarece fiecare utilizator are o bandă de frecvență proprie, nu există interferențe între utilizatori. Atunci când există doar un număr mic și constant de utilizatori, fiecare având un trafic încărcat (și bazat pe utilizarea zonelor tampon), cum ar fi, de exemplu, oficiile de comutare ale companiilor de telecomunicație, FDM este un mecanism de alocare simplu și eficient.

Cu toate acestea, atunci când numărul emițătorilor este mare și variază în permanență, sau când traficul este de tip rafală, FDM prezintă câteva probleme. Dacă spectrul benzii este împărțit în N regiuni și sunt mai puțin de N utilizatori care vor să comunice, o bună parte din bandă se va risipi. Dacă sunt mai mult de N utilizatori care vor să comunice, unii dintre ei nu o vor putea face, din lipsă de spațiu în banda de transmisie, chiar dacă există utilizatori care au primit câte o parte din bandă și transmit sau recepționează mesaje extrem de rar.

Chiar dacă presupunem că numărul utilizatorilor ar putea fi menținut în vreun fel constant la valoarea N, divizarea singurului canal disponibil în subcanale statice este, evident, ineficientă. Principala problemă este că atunci când unii utilizatori sunt inactivi, bucata lor de bandă se pierde pur și simplu. Ei nu o folosesc, dar nici alții nu au voie să o utilizeze. Mai mult, în majoritatea sistemelor de calcul, traficul de date este extrem de diferențiat (sunt uzuale raporturi de 1000:1 între traficul de vârf și cel mediu). În consecință, majoritatea canalelor vor fi libere în cea mai mare parte a timpului.

Performanțele slabe ale alocării FDM statice pot fi ușor observate dintr-un simplu calcul făcut cu ajutorul teoriei cozilor. Să luăm, pentru început, întârzierea medie, T, pentru un canal cu capacitatea C bps, la o rată a sosirilor de λ cadre/sec. Fiecare cadru are o lungime dată de o funcție de densitate de probabilitate exponențială cu media de $1/\mu$ biți/cadru. Cu acești parametri, rata sosirilor este λ cadre/sec, iar viteza de servire este μC cadre-sec. Din teoria cozilor poate fi demonstrat că pentru timpi Poisson de sosire și de servire, vom avea

$$T = \frac{1}{\mu C - \lambda}$$

De exemplu, dacă C este 100 Mbps, lungimea medie a cadrului, 1/μ, este de 10.000 de biți, iar rata sosirilor, λ, este 5.000 cadre/sec, atunci T = 200 μs. Trebuie remarcat că dacă ignoram întârzierea dată de teoria cozilor și am fi vrut să determinăm doar cât timp va dura să trimitem un cadru de 10.000 de biți într-o rețea de 100 Mbps, am fi obținut răspunsul (incorect) de 100 μs. Rezultatul, astfel calculat, este corect doar când nu există competiție pentru canal.

Acum să divizăm canalul în *N* subcanale independente, fiecare cu o capacitate de *C*/*N* bps. Rata medie a intrărilor pe fiecare subcanal va fi acum de λ/*N*. Recalculând *T* vom obține:

$$T_{FDM} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \tag{4-1}$$

Întârzierea medie la FDM este de *N* ori mai mare decât în cazul în care toate cadrele ar fi fost, printr-o scamatorie, aranjate în ordine într-o mare coadă centrală.

Exact aceeași logică utilizată la FDM se poate aplica și la multiplexarea cu diviziunea timpului (TDM - Time Division Multiplexing). Fiecărui utilizator îi este alocată static fiecare a *N*-a cuantă. Dacă un utilizator nu își folosește timpul alocat, acesta rămâne nefolosit. Același lucru se întâmplă și dacă divizăm rețelele în mod fizic. Revenind la exemplul anterior, dacă am înlocui rețeaua de 100 Mbps cu zece rețele de 10 Mbps fiecare și dacă am aloca static fiecăreia câte un utilizator, întârzierea medie ar sări de la 200 μs la 2 ms.

Deoarece nici una dintre metodele statice de alocare a canalului nu funcționează bine în condiții de trafic în rafală, vom studia în continuare metodele dinamice.

4.1.2 Alocarea dinamică a canalului în rețelele LAN și MAN

Înainte de a începe prezentarea numeroaselor metode de alocare a canalului, care fac obiectul acestui subcapitol, merită să formulăm cu atenție problema alocării. La baza întregii activități din acest domeniu stau câteva ipoteze-cheie, descrise în continuare.

1. **Modelul stațiilor**. Acest model constă din *N* **stații** independente (calculatoare, telefoane, dispozitive de comunicare personală etc.), fiecare având un program sau un utilizator care generează cadre de transmis. Stațiile sunt uneori denumite terminale. Probabilitatea de generare a unui cadru într-un interval de lungime Δt este $\lambda \Delta t$, unde λ este o constantă (rata sosirilor de cadre noi). Odată ce a fost generat un cadru, stația se blochează și nu mai face nimic până la transmiterea cu succes a cadrului.
2. **Ipoteza canalului unic**. Există un singur canal accesibil pentru toate comunicațiile. Toate stațiile pot transmite prin el și pot recepționa de la el. În ceea ce privește partea de hardware, toate stațiile sunt echivalente, deși protocolul software le poate acorda priorități diferite.
3. **Ipoteza coliziunii**. Dacă două cadre sunt transmise simultan, ele se suprapun, iar semnalul rezultat va fi neinteligibil. Acest eveniment se numește **coliziune**. Toate stațiile pot detecta coliziuni. Un cadru care a intrat în coliziune cu un alt cadru trebuie retransmis ulterior. Nu există alte erori în afara celor generate de coliziuni.
4. **Timp continuu**. Transmisia cadrelor poate surveni în orice moment. Nu există un ceas comun, care să împartă timpul în intervale discrete.

5. **Timp discret**. Timpul este împărțit în intervale discrete (cuante). Transmisia cadrelor pornește întotdeauna la începutul unei cuante. O cuantă poate conține 0, 1, sau mai multe cadre, corespunzător unei cuante de așteptare, unei transmisiuni efectuate cu succes sau, respectiv, unei coliziuni.
6. **Detecția purtătoarei**. Stațiile pot afla dacă un canal este liber sau nu înainte de a încerca să-l utilizeze. Dacă el este deja ocupat, nici o stație nu va mai încerca să îl utilizeze până când nu se va elibera.
7. **Nedetecția purtătoarei**. Stațiile nu pot afla starea canalului înainte de a încerca să îl utilizeze. Ele pur și simplu încep să transmită. Abia după aceea vor putea determina dacă transmisia s-a efectuat cu succes sau nu.

Este momentul să discutăm puțin despre aceste ipoteze. Prima dintre ele spune că stațiile sunt independente, iar cadrele sunt generate cu o frecvența constantă. De asemenea, se presupune implicit că transmisia fiecărei stații este controlată de un singur program sau de un singur utilizator, deci atâta timp cât stația este blocată, ea nu va genera noi cadre. Modelele mai sofisticate permit existența stațiilor multiprogramate, care pot genera noi cadre în timp ce stația este blocată, dar analiza acestor stații este mult mai complexă.

Ipoteza canalului unic este de fapt inima problemei. Nu există mijloace externe de comunicare. Stațiile nu pot ridica mâinile pentru a cere profesorului permisiunea de a vorbi.

Ipoteza coliziunii este, de asemenea, o ipoteză de bază, deși în unele sisteme (între care remarcăm sistemele cu spectru larg de transmisie) ea este relaxată, cu rezultate surprinzătoare. De asemenea, unele LAN-uri, cum ar fi cele de tip token-ring, utilizează un mecanism de eliminare a conflictelor, care elimină coliziunile.

Există două ipoteze alternative despre timp. Într-una din ele timpul este continuu, iar în cealaltă este discret. Unele sisteme consideră timpul într-un fel, altele în celălalt fel, așa că le vom discuta și analiza pe amândouă. Evident, pentru un sistem dat, numai una dintre ipoteze este valabilă.

În mod similar, o rețea poate avea sau nu facilități de detecție a purtătoarei. Rețelele LAN au în general detecție de purtătoare, dar rețelele prin satelit nu (datorită întârzierii mari de propagare). Stațiile din rețelele cu detecție de purtătoare își pot termina transmisia prematur, dacă descoperă că au intrat în coliziune cu o altă transmisie. De notat că aici înțelesul cuvântului „purtătoare” se referă la semnalul electric de pe cablu și nu are nimic de a face cu vreun alt tip de purtătoare.

4.2 PROTOCOALE CU ACCES MULTIPLU

Sunt cunoscuți mulți algoritmi de alocare a unui canal cu acces multiplu. În secțiunile care urmează vom studia un eșantion reprezentativ al celor mai interesanți algoritmi și vom da exemple de utilizare a lor.

4.2.1 ALOHA

În anii '70, Norman Abramson și colegii săi de la Universitatea din Hawaii au elaborat o nouă și elegantă metodă de rezolvare a problemei alocării canalului. De atunci, munca lor a fost continuată de mulți cercetători (Abramson, 1985). Deși realizarea lui Abramson, numită sistemul ALOHA,

utiliza difuzarea prin radio de la sol, ideea de bază se poate aplica la orice sistem în care utilizatori ce nu pot fi localizați concurează la utilizarea unui unic canal partajat.

Vom discuta două versiuni ale protocolului ALOHA: ALOHA pur și ALOHA cuantificat. Ele diferă prin faptul că timpul este sau nu divizat în intervale discrete, în care trebuie să se potrivească orice cadru. ALOHA pur nu cere sincronizare de timp globală, pe când ALOHA cuantificat cere.

ALOHA pur

Ideea de bază într-un sistem ALOHA este simplă: utilizatorii sunt lăsați să transmită ori de câte ori au date de trimis. Bineînțeles că vor exista coliziuni, iar cadrele intrate în coliziune vor fi distruse. Oricum, datorită proprietății de reacție a difuzării, un emițător poate afla oricând dacă mesajul său a fost distrus, ascultând canalul, la fel ca și ceilalți utilizatori. Într-o rețea LAN, reacția este imediată; într-o rețea prin satelit, există o întârziere de 270 ms înainte ca emițătorul să afle dacă transmisia s-a încheiat cu succes. În cazul în care cadrul trimis a fost distrus, emițătorul așteaptă un interval oarecare de timp și îl trimite din nou. Timpul de așteptare trebuie să fie aleatoriu, altfel aceleași cadre vor intra în coliziune iar și iar, blocându-se reciproc la nesfârșit. Sistemele în care mai mulți utilizatori partajează un canal comun într-un mod care poate duce la conflicte sunt cunoscute sub numele de **sisteme cu conflicte** (**contention systems**).

În fig. 4-1 este prezentată o schiță de generare a cadrelor într-un sistem ALOHA. Am ales să reprezentăm cadre de aceeași lungime, pentru că productivitatea sistemelor ALOHA este maximizată în cazul în care avem cadre de lungime uniformă, față de cazul în care avem cadre de lungime variabilă.

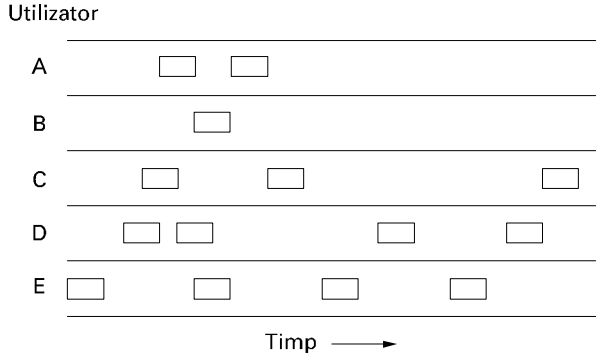


Fig. 4-1. În ALOHA pur, cadrele sunt transmise la momente complet arbitrare.

Ori de câte ori două cadre încearcă să ocupe canalul în același timp, se va produce o coliziune și amândouă vor fi denaturate. Dacă primul bit al unui nou cadru se suprapune cu ultimul bit al unui cadru aproape terminat, amândouă cadrele vor fi total distruse și amândouă vor trebui retransmise mai târziu. Suma de control nu poate (și nu trebuie) să distingă între o pierdere totală și o ratare „la mustață”. Ceea ce este rău este rău.

O întrebare foarte interesantă este: care este eficiența unui canal ALOHA? Cu alte cuvinte, ce fracțiune din cadrele transmise nu intră în coliziune în aceste circumstanțe haotice? Să considerăm mai întâi o colectivitate infinită de utilizatori interactivi stând în fața calculatoarelor (stațiilor) lor. Un utilizator este întotdeauna într-una din cele două stări: introduce caractere sau așteaptă. Inițial, toți utilizatorii sunt în prima stare, scriind. Când termină o linie, utilizatorul se oprește din scris, așteptând un răspuns. Atunci stația transmite pe canal un cadru conținând linia și verifică dacă trans-

misia s-a efectuat cu succes. Dacă da, utilizatorul vede răspunsul și se apucă din nou de scris. Dacă nu, utilizatorul continuă să aștepte, iar cadrul va fi transmis în mod repetat, până când transmisia se va încheia cu succes.

Să numim „interval de cadru" timpul necesar pentru a transmite un cadru standard, de lungime fixă (adică lungimea cadrului împărțită la rata biților). Vom presupune că populația infinită de utilizatori generează cadre noi conform unei distribuții Poisson cu media de N cadre pe interval de cadru (ipoteza populației infinite este necesară pentru a ne asigura că N nu descrește pe măsură ce utilizatorii se blochează). Dacă $N > 1$, utilizatorii generează cadre cu o rată mai mare decât capacitatea de transmisie a canalului și aproape fiecare cadru va suferi o coliziune. Pentru o productivitate rezonabilă ar trebui ca $0 < N < 1$.

În plus față de noile cadre, stațiile mai generează și copii ale cadrelor care au suferit anterior coliziuni. Să presupunem în continuare că probabilitatea de a avea k încercări de transmisie pe interval de cadru, incluzând și retransmisiile, are de asemenea o distribuție Poisson, cu media G pe interval de cadru. Evident, $G \geq N$. La încărcare redusă (adică $N \approx 0$), vor fi puține coliziuni, deci puține retransmisii, așa că $G \approx N$. La încărcare mare vor fi multe coliziuni, deci $G > N$. Orice încărcare am avea, productivitatea este chiar încărcarea dată, G, înmulțită cu probabilitatea ca o transmisie să se încheie cu succes - adică $S = GP_0$, unde P_0 este probabilitatea ca un cadru să nu sufere coliziuni.

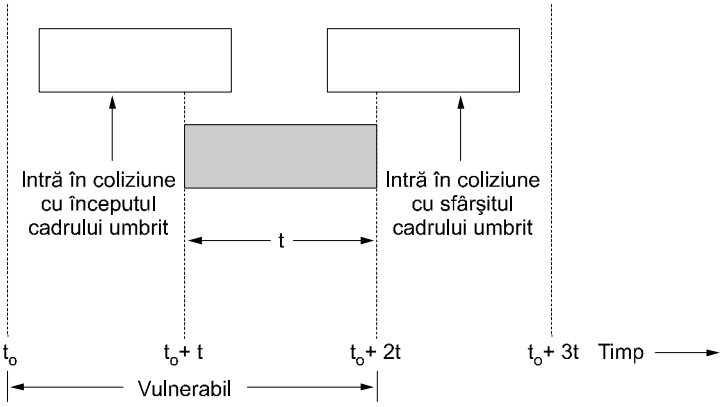


Fig. 4-2. Perioada vulnerabilă pentru cadrul umbrit.

Un cadru nu va suferi coliziuni dacă nici un alt cadru nu va fi emis în intervalul de un cadru socotit de la începutul lui, așa cum se arată în fig. 4-2. În ce condiții cadrul umbrit va ajunge întreg? Fie t timpul necesar emisiei unui cadru. Dacă un alt utilizator a generat un cadru între t_0 și $t_0 + t$, sfârșitul acelui cadru va intra în coliziune cu începutul cadrului umbrit. De fapt, soarta cadrului umbrit era deja pecetluită chiar înainte de transmisia primului bit, dar cum în ALOHA pur, o stație nu ascultă canalul înainte de transmisie, nu are cum să știe că un alt cadru se află deja în curs de transmisie. Similar, orice alt cadru care începe între $t_0 + t$ și $t_0 + 2t$ va nimeri peste sfârșitul cadrului umbrit.

Probabilitatea ca într-un interval de cadru dat să fie generate un număr k de cadre este modelată de distribuția Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{4-2}$$

deci probabilitatea generării a zero cadre este doar e^{-G}. Într-o perioadă de timp cât două intervale de cadru, media numărului de cadre generate este $2G$. Astfel, probabilitatea ca nici o transmisie să nu înceapă în timpul perioadei de timp vulnerabile este dată de $P_0 = e^{-2G}$.
Luând $S = GP_0$, obținem:

$$S = Ge^{-2G}$$

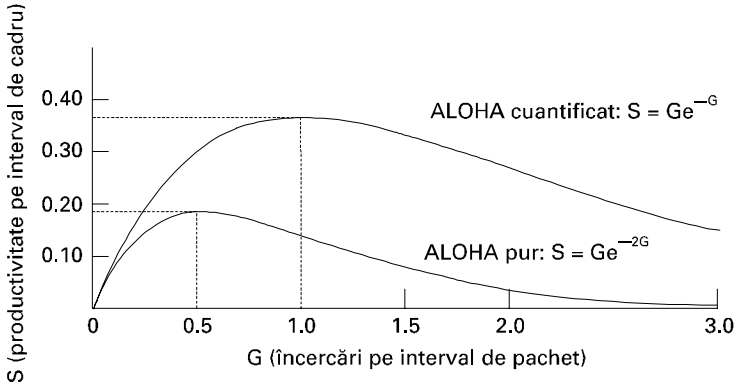


Fig. 4-3. Productivitatea în funcție de traficul oferit pentru sistemele ALOHA.

Relația dintre traficul oferit și productivitate este prezentată în fig. 4-3. Productivitatea maximă este obținută la $G = 0.5$, cu $S = 1/2e$, adică aproximativ 0.184. Cu alte cuvinte, cea mai bună performanță la care putem spera este o utilizare a canalului de 18 procente. Acest rezultat nu este prea încurajator, dar, în situația aceasta în care fiecare utilizator transmite la dorință, cu greu ne-am fi putut aștepta la o performanță de sută la sută.

ALOHA cuantificat

În 1972, Roberts a publicat o metodă de dublare a capacității unui sistem ALOHA (Roberts, 1972). Propunerea lui era să se împartă timpul în intervale discrete, fiecare interval corespunzând unui cadru. Această abordare cere ca utilizatorii să cadă de acord asupra mărimii cuantelor. O cale de a obține sincronizarea ar fi ca o stație specială să emită un „bip" la începutul fiecărui interval, ca un tact de ceas.

În metoda lui Roberts, care a devenit cunoscută sub numele de **ALOHA cuantificat** (**slotted ALOHA**), în contrast cu metoda lui Abramson - **ALOHA pur** (**pure ALOHA**), unui calculator nu îi este permis să emită ori de câte ori este apăsată tasta „Return". El este nevoit să aștepte începutul următoarei cuante. Astfel, protocolul ALOHA pur este transformat din continuu în discret. Deoarece acum perioada vulnerabilă este înjumătățită, probabilitatea ca în intervalul cadrului nostru de test să nu mai apară un alt trafic este e^{-G}, ceea ce conduce la:

$$S = Ge^{-G} \tag{4-3}$$

Așa cum reiese din fig. 4-3, ALOHA cuantificat prezintă un maxim la $G = 1$, cu o productivitate de $S = 1/e$, adică aproximativ 0.368, dublu față de ALOHA pur. Dacă sistemul operează la $G = 1$, probabilitatea unei cuante neutilizate este 0.368 (din ecuația 4-2). Cea mai bună performanță la care ne putem aștepta de la ALOHA cuantificat este: 37% din cuante neutilizate, 37% cadre transmise cu succes și 26% coliziuni. Lucrul cu valori mai mari ale lui G reduce numărul cuantelor neutilizate, dar îl mărește exponențial pe cel al coliziunilor. Pentru a vedea cum se explică rapida creștere a nu-

mărului coliziunilor odată cu G, să considerăm transmisia unui cadru de test. Probabilitatea ca el să evite o coliziune este e^{-G}, adică probabilitatea ca toți ceilalți utilizatori să nu transmită în acest interval. Probabilitatea unei coliziuni este deci $1 - e^{-G}$. Probabilitatea ca o transmisie să se efectueze exact din k încercări (adică după k - 1 coliziuni, urmate de un succes) este

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

Numărul de transmisii prognozat pentru fiecare apăsare a tastei „Return", E, este deci

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

Ca urmare a dependenței exponențiale a lui E față de G, creșteri mici ale încărcării canalului pot reduce drastic performanțele sale.

ALOHA cuantificat este important pentru un motiv care poate nu este evident la prima vedere. A fost conceput în anii `70, a fost folosit în câteva sisteme experimente timpurii, apoi a fost aproape uitat. Când a fost inventat accesul la Internet prin cablu a apărut dintr-o dată problema alocării unui singur canal între utilizatori multipli aflați în concurență – astfel încât ALOHA cuantificat a fost scos de la naftalină pentru a salva situația. S-a întâmplat frecvent ca protocoale perfect valide să fie date uitării din motive politice (de exemplu, deoarece o mare companie vrea ca toată lumea să facă lucrurile așa cum zice ea), dar la ani distanță o persoană inteligentă descoperă că un protocol de mult uitat rezolva o anumită problemă curentă. Din acest motiv, în acest capitol vom studia un număr de protocoale elegante care nu sunt actualmente folosite la scară largă, dar care ar putea foarte simplu să fie utile în aplicațiile viitorului, dacă suficienți ingineri de rețea sunt conștienți de existența lor. Desigur, vom studia mai multe protocoale aflate în folosință curentă.

4.2.2 Protocoale cu acces multiplu și detecție de purtătoare

Cu ALOHA cuantificat se poate atinge un grad de utilizare a canalului de până la $1/e$. Acest lucru nu este surprinzător, dacă ne gândim că stațiile transmit când doresc, fără a fi atente la ceea ce fac celelalte stații și, în consecință, vor exista numeroase coliziuni. Oricum, în rețelele locale, stațiile pot detecta ce fac celelalte stații și își pot adapta comportamentul în mod corespunzător. Astfel de rețele pot obține un grad de utilizare mult mai bună decât $1/e$. În această secțiune vom discuta câteva protocoale pentru îmbunătățirea a performanței.

Protocoalele în care stațiile ascultă pentru a detecta o purtătoare (adică o transmisie) și acționează corespunzător se numesc **protocoale cu detecție de purtătoare** (**carrier sense protocols**). Kleinrock și Tobagi (1975) au analizat în detaliu câteva protocoale de acest tip. În continuare vom prezenta câteva versiuni ale protocoalelor cu detecție de purtătoare.

CSMA persistent și nepersistent

Primul protocol cu detecție de purtătoare pe care îl vom studia în acest material se numește **CSMA 1-persistent** (**Carrier Sense Multiple Access**, rom: acces multiplu cu detecție de purtătoare). Atunci când o stație are date de transmis, mai întâi ascultă canalul pentru a vedea dacă nu cumva transmite altcineva în acel moment. În cazul în care canalul este ocupat, stația așteaptă până la eliberarea sa. Atunci când stația detectează canalul liber, transmite un cadru. Dacă se produce o coliziune, stația așteaptă o perioadă aleatorie de timp și o ia de la început. Protocolul se cheamă 1-persistent, pentru că probabilitatea ca o stație să transmită atunci când găsește canalul liber este egală cu 1.

Întârzierea de propagare are o influență importantă asupra performanței protocolului. Există o oarecare șansă ca, imediat după ce o stație începe să transmită, o altă stație să devină pregătită de transmisie și să asculte canalul. Dacă semnalul primei stații nu a ajuns încă la cea de-a doua, aceasta din urmă va detecta canalul liber și va începe la rândul ei să emită, rezultând o coliziune. Cu cât este mai mare întârzierea de propagare, cu atât acest efect devine mai important, iar performanța protocolului scade.

Chiar dacă întârzierea de propagare ar fi zero, tot s-ar mai produce coliziuni. Dacă două stații devin gata de transmisie în timpul transmisiunii unei a treia stații, amândouă vor aștepta politicos până la sfârșitul ei, după care vor începe să transmită simultan, producându-se o coliziune. Dacă ele nu ar fi atât de nerăbdătoare, s-ar produce mai puține coliziuni. Chiar și așa, acest protocol este semnificativ mai bun decât ALOHA pur, întrucât ambele stații au bunul simț să nu interfereze cu cadrul celei de-a treia stații. Intuitiv, acest fapt va conduce la o performanță mai bună decât ALOHA pur. Același lucru este valabil și pentru ALOHA cuantificat.

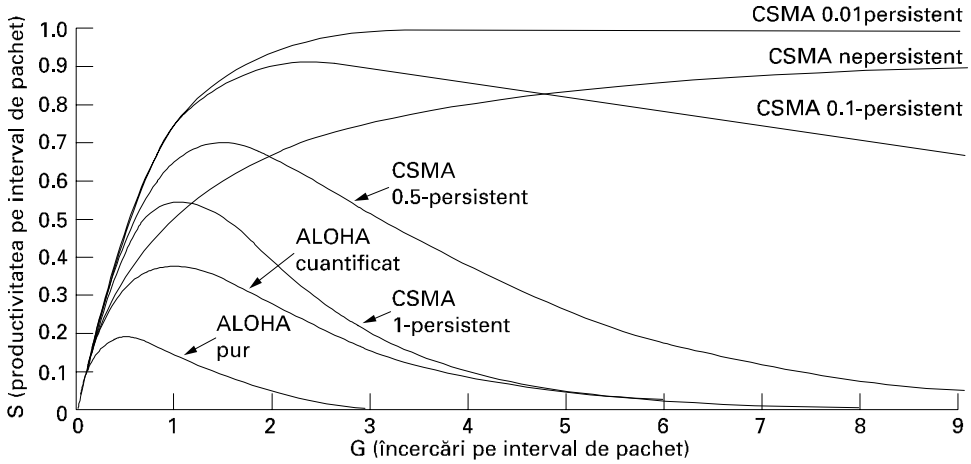


Fig. 4-4. Comparație între utilizările canalului în funcție de încărcare, pentru diferite protocoale cu acces aleator.

Un al doilea protocol cu detecție de purtătoare este **CSMA nepersistent** (**nonpersistent CSMA**). În acest protocol, o încercare conștientă de transmisie este mai puțin „lacomă" decât în cel anterior. Înainte de a emite, stația ascultă canalul. Dacă nimeni nu emite, începe ea să emită. Dacă însă canalul este ocupat, stația nu rămâne în continuu în ascultare, pentru a-l ocupa imediat după detectarea sfârșitului transmisiei precedente. În schimb, așteaptă o perioadă aleatorie de timp și apoi repetă algoritmul. Intuitiv, acest algoritm ar trebui să conducă la o utilizare mai bună a canalului, dar și la întârzieri mai mari decât la CSMA 1-persistent.

Ultimul protocol este **CSMA p-persistent** (**p-persistent CSMA**). El se aplică la canalele cuantificate și funcționează după cum urmează. Când o stație este gata să emită, ea ascultă canalul. Dacă acesta este liber, stația va transmite cu o probabilitate p. Cu probabilitatea $q = 1 - p$, stația va aștepta următoarea cuantă. Dacă această cuantă este de asemenea liberă, va transmite sau va aștepta din nou, cu probabilitățile p și respectiv q. Acest proces este repetat până când cadrul este transmis sau până când o altă stație începe să transmită. În ultimul caz, stația se comportă ca și când s-ar fi produs o coliziune (adică așteaptă o perioadă aleatorie de timp și pornește iar). Dacă inițial stația detectează canalul ocupat, așteaptă cuanta următoare și aplică algoritmul de mai sus. Fig. 4-4

ză canalul ocupat, așteaptă cuanta următoare și aplică algoritmul de mai sus. Fig. 4-4 arată productivitatea în funcție de traficul oferit pentru toate cele trei protocoale, precum și pentru ALOHA pur și ALOHA cuantificat.

CSMA cu detecția coliziunii

Protocoalele CSMA persistent și nepersistent reprezintă în mod cert o îmbunătățire față de ALOHA, pentru că au grijă ca nici o stație să nu înceapă să transmită atunci când canalul este ocupat. O altă îmbunătățire este abandonarea transmisiei îndată ce se detectează o coliziune. Cu alte cuvinte, dacă două stații găsesc canalul liber și încep să transmită simultan, amândouă vor detecta coliziunea aproape imediat. Decât să își termine de transmis cadrele, care oricum sunt iremediabil denaturate, stațiile își vor termina brusc transmisia imediat după detectarea coliziunii. Terminând repede cu cadrele distruse, se salvează timp și lărgime de bandă. Acest protocol, cunoscut sub numele de **CSMA/CD** (**Carrier Sense Multiple Access with Collision Detection**, rom: acces multiplu cu detecția purtătoarei și a coliziunii), este des întrebuințat în LAN-uri în subnivelul MAC. În particular, este baza popularului Ethernet LAN, astfel încât merită efortul să îl analizăm în detaliu.

CSMA/CD, ca și multe alte protocoale de LAN, utilizează modelul conceptual din fig. 4-5. În momentul marcat cu t_0, o stație oarecare își termină de transmis cadrul. Acum orice altă stație care are de transmis un cadru poate încerca să transmită. Dacă două sau mai multe stații se decid să transmită simultan, se va produce o coliziune. Coliziunile pot fi detectate urmărind puterea sau lățimea impulsului semnalului recepționat și comparându-le cu semnalul transmis.

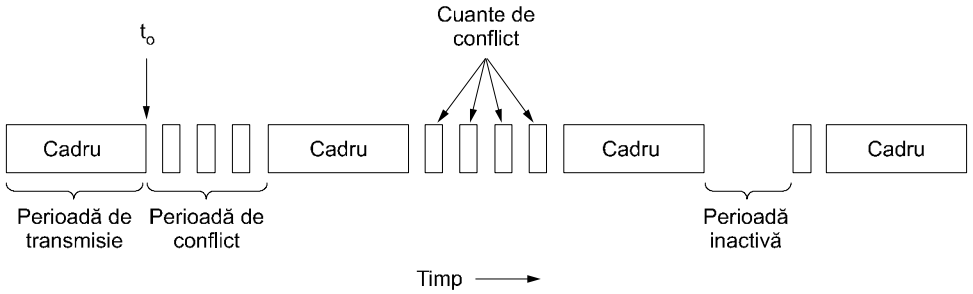


Fig. 4-5. CSMA/CD se poate afla într-una din următoarele stări: conflict, transmisie sau inactiv.

După ce o stație a detectat o coliziune, își abandonează transmisia, așteaptă o perioadă de timp oarecare și încearcă din nou, dacă nici o altă stație nu a început să transmită între timp. De aceea, modelul nostru pentru CSMA/CD va fi alcătuit alternativ din perioade de timp cu transmisii și perioade de timp de conflict, având și perioade de așteptare, când toate stațiile tac (de exemplu, din lipsă de activitate).

Să privim acum mai îndeaproape detaliile algoritmului de tratare a conflictelor. Să presupunem că două stații încep să transmită simultan, exact la momentul t_0. Cât timp le va lua ca să-și dea seama că s-a produs o coliziune? Răspunsul la această întrebare este vital pentru determinarea mărimii perioadei de conflict, deci și a întârzierii și a productivității. Timpul minim de detectare a coliziunii este chiar timpul necesar propagării semnalului de la o stație la alta.

Bazându-ne pe acest raționament, am putea crede că o stație care nu detectează nici o coliziune într-o perioadă de timp egală cu timpul de propagare pe toată lungimea cablului, perioadă măsurată de la începutul transmisiei, poate fi sigură că a ocupat canalul. Prin „ocupat” înțelegem că toate cele-

lalte stații știu că ea transmite și nu vor interfera cu ea. Această concluzie este greșită. Să considerăm cazul cel mai defavorabil, descris în următorul scenariu. Fie τ timpul de propagare a semnalului între stațiile cele mai îndepărtate. La t_0, o stație începe să transmită. La $t_0 + (\tau - \varepsilon)$, cu o clipă înainte ca semnalul să ajungă la cea mai îndepărtată stație, aceasta începe la rândul ei să transmită. Bineînțeles, ea detectează coliziunea aproape instantaneu și se oprește, dar scurta rafală de zgomot produsă de coliziune nu se va întoarce la stația de origine decât după $2\tau - \varepsilon$. Cu alte cuvinte, în cel mai rău caz, o stație nu poate fi sigură că a ocupat canalul decât după ce a transmis timp de 2τ fără a detecta vreo coliziune. Din acest motiv vom modela intervalul de conflict ca un sistem ALOHA cuantificat, cu dimensiunea cuantei 2τ. Pe un cablu coaxial de 1 km, $\tau \approx 5$ μs. Simplificând, vom presupune că fiecare cuantă conține un singur bit. Bineînțeles că, odată ce canalul a fost ocupat, o stație poate transmite cu orice rată dorește, nu neapărat doar 1 bit la 2τ sec.

Este important să înțelegem că detecția coliziunii este un proces *analogic*. Echipamentul stației trebuie să asculte cablul în timp ce transmite. Dacă ceea ce recepționează este diferit față de ceea ce transmite, înseamnă că se produce o coliziune. Așadar, codificarea semnalului trebuie să permită detectarea coliziunilor (de exemplu, o coliziune a două semnale de 0 volți poate fi imposibil de detectat). Din acest motiv, de obicei se utilizează codificări speciale.

Este de asemenea important de observat că o stație care transmite trebuie să monitorizeze continuu semnalul, să asculte zgomotele care pot indica o coliziune. Din acest motiv, CSMA/CD cu un singur canal este în mod inerent un sistem jumătate-duplex (half-duplex). Este imposibil ca o stație să transmită și să primească cadre simultan datorită faptului că logica primirii cadrelor este activă, verificând coliziunile în timpul fiecărei transmisiuni.

Pentru a evita orice neînțelegere, e bine să notăm că nici un protocol al subnivelului MAC nu garantează o livrare corectă a cadrelor. Chiar și în absența coliziunilor, receptorul poate să nu fi copiat corect cadrul din diverse motive (de exemplu, lipsă de spațiu în zona tampon, sau o întrerupere ratată).

4.2.3 Protocoale fără coliziuni

Deși în CSMA/CD nu apar coliziuni după ce o stație a ocupat efectiv canalul, ele mai pot apărea în perioada de conflict. Aceste coliziuni afectează negativ performanța sistemului, mai ales atunci când cablul este lung (adică τ mare), iar cadrele scurte. Pe măsură ce rețelele bazate pe fibre optice foarte lungi și cu lărgime mare de bandă sunt tot mai folosite, combinația de valori mari pentru τ și cadre scurte va deveni o problemă din ce în ce mai serioasă. În această secțiune vom examina câteva protocoale care rezolvă conflictul pentru canal fără nici o coliziune, nici măcar în perioada de conflict.

În protocoalele ce vor fi descrise în continuare, vom presupune că există N stații, fiecare având o adresă unică fixă, cuprinsă între 0 și $N - 1$. Nu contează dacă unele stații sunt inactive o parte din timp. Întrebarea de bază rămâne: care stație va primi canalul după o transmisie efectuată cu succes? Vom continua să folosim modelul din fig. 4-5, cu cuantele sale discrete de conflict.

Protocolul Bit-Map (cu hartă de biți)

În primul nostru protocol fără coliziuni, **metoda bit-map de bază** (**basic bit-map method**), fiecare perioadă de conflict va fi formată din exact N cuante. Dacă stația 0 are de transmis un cadru, transmite un bit 1 în timpul cuantei 0. Nici o altă stație nu are voie să transmită în timpul acestei cuante. Fără a avea vreo legătură cu ceea ce face stația 0, stația 1 are ocazia să transmită un 1 în timpul cuantei 1, dar doar dacă are un cadru de transmis. În general, stația j poate anunța că are de transmis un

cadru inserând un bit 1 în cuanta *j*. După ce au trecut toate cele *N* cuante, fiecare stație va cunoaște care dintre stații doresc să transmită. În acest moment, ele încep să transmită în ordinea crescătoare a adresei de stație (vezi fig. 4-6).

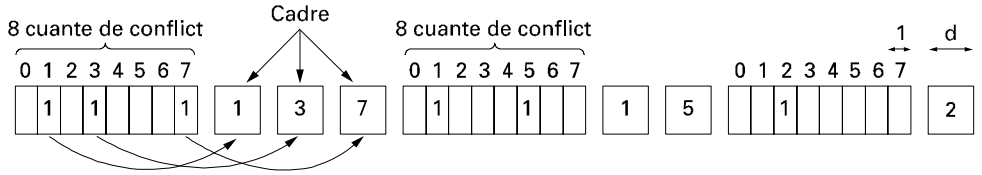


Fig. 4-6. Protocolul bit-map de bază.

Întrucât toți sunt de acord cine urmează, nu vor exista niciodată coliziuni. După ce ultima dintre stațiile pregătite să emită și-a transmis cadrul, eveniment pe care toate stațiile îl pot urmări ușor, va începe o altă perioadă de conflict de *N* biți. Dacă o stație devine gata imediat după ce a trecut cuanta care îi corespunde, înseamnă că a ratat ocazia și va trebui să aștepte următoarea perioadă de conflict. Protocoalele de acest gen, în care intenția de a transmite este anunțată înainte de transmisia propriu-zisă, se numesc **protocoale cu rezervare** (reservation protocols).

Să analizăm pe scurt performanțele acestui protocol. Vom conveni ca timpul să fie măsurat în unități de mărimea cuantelor de un bit ale perioadei de conflict, iar cadrele de date să fie formate din *d* astfel de unități de timp. Practic, în condiții de trafic slab, pachetul de biți ai perioadei de conflict va fi transmis în mod repetat, din lipsă de cadre de date.

Să privim situația din punctul de vedere al unei stații cu adresă mică, de exemplu 0 sau 1. În mod obișnuit, când ea devine gata să emită, cuanta „curentă" va fi undeva în mijlocul pachetului de biți. În medie, o stație va trebui să aștepte $N/2$ cuante pentru ca runda curentă să se termine și alte N cuante până la runda următoare, înainte de a putea începe transmisia.

Perspectivele stațiilor cu adrese mari sunt ceva mai luminoase. În general, ele nu vor trebui să aștepte decât o jumătate de rundă ($N/2$ cuante) înainte de a începe să transmită. Stațiile cu adrese mari trebuie rareori să aștepte următoarea rundă. Deoarece stațiile cu adrese mici au de așteptat în medie $1.5N$ cuante, iar cele cu adrese mari $0.5N$ cuante, media pentru toate stațiile este de N cuante. Eficiența canalului la trafic scăzut este ușor de calculat. Încărcarea suplimentară a unui cadru este de N biți, iar cantitatea de date este de d biți, rezultând o eficiență de $d/(N + d)$.

În condiții de trafic încărcat, când toate stațiile vor să emită simultan, perioada de conflict de N biți este împărțită la N cadre, rezultând o încărcare suplimentară de doar un bit pe cadru, adică o eficiență de $d/(d + 1)$. Întârzierea medie pentru un cadru este egală cu suma timpului de așteptare în interiorul stației, plus o întârziere suplimentară de $N(d + 1)/2$, care se adaugă atunci când ajunge la începutul cozii interne a stației.

Numărătoarea inversă binară

O problemă a protocolului de bază bit-map este încărcarea suplimentară de 1 bit pe stație. Putem obține rezultate și mai bune utilizând adresele binare ale stațiilor. O stație care vrea să utilizeze canalul își difuzează adresa ca un șir de biți, începând cu bitul cel mai semnificativ. Se presupune că toate adresele au aceeași lungime. Biții de pe aceeași poziție din adresele diferitelor stații sunt combinați printr-o operație logică OR (SAU), iar rezultatul este citit ca o singură adresă. Vom numi acest protocol **numărătoarea inversă binară** (**binary countdown**). El este utilizat în Datakit (Fraser, 1987). Se presupune implicit că întârzierile în transmisie sunt neglijabile, astfel încât toate stațiile văd biții transmiși practic instantaneu.

Pentru a evita conflictele, trebuie aplicată o regulă de arbitrare: de îndată ce o stație observă că unul dintre biții superiori ai adresei sale, conținând un 0, a fost acoperit de un 1, renunță să mai emită. De exemplu, dacă stațiile 0010, 0100, 1001 și 1010 încearcă să obțină canalul în același timp, în timpul primului bit stațiile transmit 0, 0, 1 și, respectiv, 1. Acești biți sunt combinați printr-o operație SAU, rezultând un 1. Stațiile 0010 și 0100 văd acest 1 și știu că o stație cu o adresă superioară încearcă să obțină canalul, așa că renunță să mai emită în runda curentă. Stațiile 1001 și 1010 continuă.

Următorul bit este 0 și ambele stații continuă. Următorul bit este 1, așa că stația 1001 va renunța. Câștigătoare este stația 1010, pentru că are adresa cea mai mare. După ce a câștigat licitația, ea poate transmite un cadru, după care începe o nouă rundă de licitații. Protocolul este ilustrat în fig. 4-7. Are proprietatea că stațiile cu numere mai mari au o prioritate mai înaltă decât stațiile cu numere mai mici, ceea ce poate fi și bine și rău, în funcție de context.

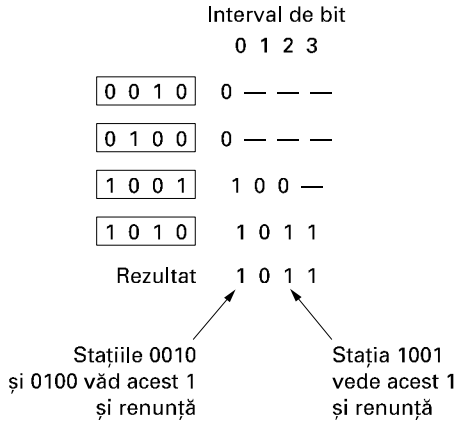


Fig. 4-7. Protocolul cu numărătoare inversă binară. O liniuță indică tăcere.

Eficiența canalului, în cazul acestei metode, este de $d/(d + \ln N)$. Dacă formatul cadrului a fost bine ales, astfel încât adresa emițătorului să fie primul câmp al cadrului, chiar și acești ln N biți nu sunt pierduți, iar eficiența este de 100%.

Mok și Ward (1979) au descris o variantă a numărătorii inverse binare utilizând o interfață paralelă în locul celei seriale. Ei au sugerat, de asemenea, utilizarea de adrese de stație virtuale, cuprinse între 0 și numărul stației câștigătoare inclusiv, adrese ce vor fi permutate după fiecare transmisie, pentru a da prioritate mai mare stațiilor care nu au mai transmis de mult. De exemplu, dacă stațiile *C, H, D, A, G, B, E* și *F* au prioritățile 7, 6, 5, 4, 3, 2, 1 și, respectiv, 0, atunci o transmisie cu succes a stației *D* o va plasa la sfârșitul listei, rezultând ordinea priorităților: *C, H, A, G, B, E, F, D*. Astfel, *C* rămâne virtual stația 7, A suie de la 4 la 5, iar *D* coboară de la 5 la 0. Acum stația *D* va putea obține canalul numai dacă nici o altă stație nu îl dorește.

Numărătoarea inversă binară este un exemplu de protocol simplu, elegant și eficient care așteaptă să fie redescoperit. Sperăm că își va găsi o nouă familie cândva în viitor.

4.2.4 Protocoale cu conflict limitat

Am considerat până acum două strategii de bază pentru obținerea canalului într-o rețea cablată: cu conflict, ca în CSMA, și fără coliziuni. Fiecare strategie poate fi cotată după performanțe în func-

ție de doi parametri importanți: întârzierea în condiții de trafic scăzut și eficiența canalului la trafic încărcat. În condițiile unui trafic scăzut, conflictul (adică ALOHA pur sau cuantificat) este preferat datorită întârzierilor mici. Cu cât traficul crește, cu atât aceste metode devin tot mai puțin atractive, deoarece încărcarea suplimentară asociată cu arbitrarea canalului devine tot mai mare. Pentru protocoalele fără coliziuni apare efectul invers: la trafic scăzut, ele au întârzieri mari, dar, pe măsură ce traficul crește, eficiența canalului se îmbunătățește în loc să se înrăutățească, cum se întâmplă la protocoalele cu conflict.

Evident, ar fi frumos să putem combina cele mai bune proprietăți ale protocoalelor cu conflict cu cele ale protocoalelor fără coliziuni, obținând un nou protocol care să utilizeze varianta cu conflict la trafic scăzut, pentru a avea întârzieri mici, și varianta fără coliziuni la trafic mare, pentru a putea oferi o eficiență bună a canalului. Asemenea protocoale, pe care le vom numi **protocoale cu conflict limitat** (**limited contention protocols**), există și vor încheia studiul nostru despre rețelele cu detecție de purtătoare.

Până acum, singurele protocoale cu conflict pe care le-am studiat au fost simetrice, adică fiecare stație încearcă să obțină canalul cu o probabilitate p, aceeași pentru toate stațiile. Un fapt destul de interesant este că performanța globală a sistemului poate fi uneori îmbunătățită utilizând un protocol care asociază probabilități diferite pentru stații diferite.

Înainte de a trece la protocoalele asimetrice, să trecem succint în revistă performanțele cazului simetric. Să presupunem că există k stații care concurează pentru obținerea accesului la canal. Fiecare are o probabilitate p de a transmite în timpul fiecărei cuante. Probabilitatea ca o stație să obțină canalul în timpul unei cuante este $kp\ (1 - p)^{k-1}$. Pentru a obține valoarea optimă pentru p, derivăm în raport cu p, egalăm rezultatul cu zero și rezolvăm pentru p. Vom obține că valoarea cea mai bună a lui p este $1/k$. Substituind $p = 1/k$, obținem probabilitatea

$$\Pr[\text{succes cu } p \text{ optim}] = \left(\frac{k-1}{k}\right)^{k-1} \tag{4-4}$$

Această probabilitate este reprezentată în fig. 4-8. Pentru un număr mic de stații șansele de succes sunt mari, dar probabilitatea scade către o valoare asimptotică de $1/e$ înainte chiar ca numărul stațiilor să atingă valoarea cinci.

Din fig. 4-8 reiese clar că probabilitatea ca o stație să obțină canalul poate fi crescută doar reducând concurența. Protocoalele cu conflict limitat fac exact acest lucru. Mai întâi, ele împart stațiile în grupuri (nu neapărat disjuncte). Doar membrilor grupului 0 li se permite să concureze pentru cuanta 0. Dacă unul din ei reușește, ocupă canalul și își transmite cadrul. În cazul în care cuanta rămâne neîntrebuințată sau apare o coliziune, membrii grupului 1 vor concura pentru cuanta 1 etc. Făcând o împărțire corectă a stațiilor în grupuri, numărul de conflicte pentru fiecare cuantă poate fi redus, aducând performanța corespunzătoare fiecărei cuante către extrema stânga a fig. 4-8.

Trucul constă în modul în care asociem stațiile cuantelor. Înainte de a analiza cazul general, să considerăm câteva cazuri particulare. La o extremă, fiecare grup are un singur membru. O astfel de împărțire garantează că niciodată nu vom avea coliziuni, pentru că cel mult o stație concurează pentru o cuantă. Am văzut astfel de protocoale anterior (de exemplu, numărătoarea inversă binară). Următorul caz particular este împărțirea în grupuri de câte două stații. Probabilitatea ca amândouă să încerce să transmită în timpul unei cuante este p^2, ceea ce pentru un p mic este o valoare neglijabilă. Pe măsură ce unei cuante îi sunt asociate mai multe stații, probabilitatea unei coliziuni crește, în schimb lungimea pachetului de biți, necesar pentru a da fiecăruia o șansă, se micșorează. Cazul limită este un singur grup conținând toate stațiile (adică ALOHA cuantificat). Ceea ce ne trebuie este o

cale de a asocia stații cuantelor în mod dinamic, cu multe stații pe cuantă atunci când traficul este scăzut și puține stații pe cuantă (sau chiar una singură) atunci când traficul este mare.

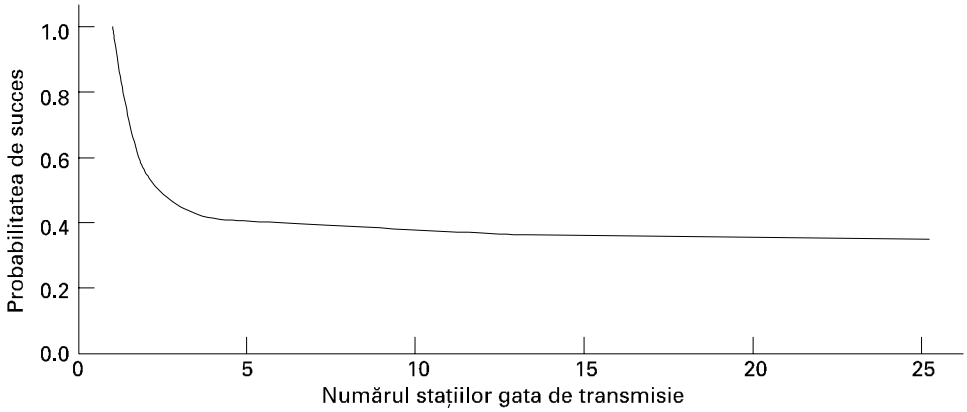


Fig. 4-8. Probabilitatea de obținere a unui canal cu conflict simetric.

Protocolul cu parcurgere arborescentă adaptivă

O cale foarte simplă de a face o asociere bună este utilizarea algoritmului conceput de armata Statelor Unite în scopul testării pentru sifilis a soldaților în timpul celui de-al doilea război mondial (Dorfman, 1943). Pe scurt, armata preleva eșantioane de sânge de la *N* soldați. O porțiune din fiecare eșantion era pusă în același tub de test. Acest eșantion mixat era apoi testat pentru anticorpi. Dacă nu era găsit nici un anticorp, toți soldații din grup erau declarați sănătoși. Dacă însă erau prezenți anticorpi, erau preparate două noi eșantioane mixte, unul corespunzător soldaților de la 1 la *N*/2, iar altul corespunzător celorlalți. Procesul era repetat recursiv până când erau determinați soldații infectați.

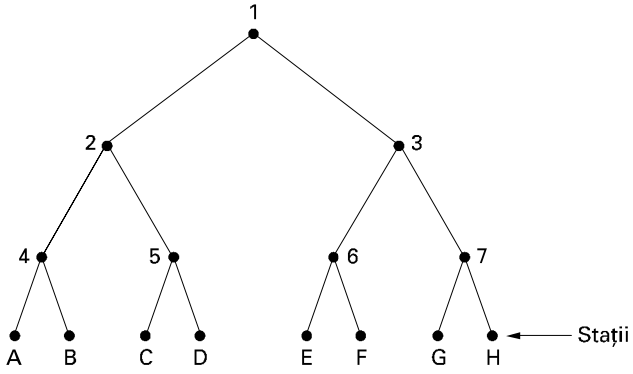


Fig. 4-9. Arborele pentru opt stații.

Pentru versiunea informatică a acestui algoritm (Capetanakis, 1979), cel mai simplu este să ne închipuim stațiile ca fiind frunzele unui arbore binar, ca în fig. 4-9. În prima cuantă de conflict care urmează după un cadru transmis cu succes, și anume cuanta 0, toate stațiile au permisiunea de a încerca ocuparea canalului. Dacă numai una din ele încearcă, foarte bine. Dacă s-a produs o coliziune, atunci, în timpul cuantei 1, doar stațiile de sub nodul 2 din arbore pot concura. Dacă una din ele obține canalul, cuanta care urmează cadrului ce va fi transmis este rezervat pentru stațiile de sub nodul 3. Dacă, pe de altă parte, două sau mai multe stații de sub nodul 2 vor să transmită, se va produce o coliziune în timpul cuantei 1, caz în care va fi rândul nodului 4 în timpul cuantei 2.

În principiu, dacă apare o coliziune în timpul cuantei 0, este cercetat întregul arbore în adâncime, pentru a localiza toate stațiile gata să transmită. Fiecare cuantă de un bit este asociată unui nod particular din arbore. Dacă se produce o coliziune, căutarea este continuată recursiv cu fiii stâng și drept ai nodului. Dacă o cuantă de un bit este liber sau dacă o singură stație transmite în timpul ei, căutarea nodului pentru această cuantă se poate opri, pentru că toate stațiile gata să transmită au fost localizate (dacă erau mai multe decât una s-ar fi produs o coliziune).

Atunci când încărcarea sistemului este mare, nu prea merită să dedicăm cuanta 0 nodului 1, pentru că acest lucru ar avea sens doar în eventualitatea - destul de puțin probabilă - ca o singură stație să aibă un cadru de transmis. Similar, se poate argumenta că nodurile 2 și 3 pot fi lăsate la o parte din aceleași motive. În termeni mai generali, întrebarea este: la ce nivel din arbore ar trebui să înceapă căutarea? Desigur, cu cât traficul este mai mare, cu atât căutarea trebuie să înceapă mai de jos. Vom considera că fiecare stație deține o estimare corectă a numărului de stații gata să transmită, de exemplu q, obținută din monitorizarea traficului recent.

Pentru început, să numărăm nivelurile arborelui începând de la vârf, cu nodul 1 din fig. 4-9 pe nivelul 0, nodurile 2 și 3 pe nivelul 1 etc. Observați că fiecare nod de pe nivelul i are dedesubt o fracțiune de 2^{-i} din totalul stațiilor. Dacă cele q stații gata să transmită sunt uniform distribuite, numărul celor care se află sub un anumit nod de pe nivelul i este $2^{-i}q$. Intuitiv, ar trebui ca nivelul optim de începere a căutării să fie cel pentru care numărul mediu de stații care vor să transmită în timpul unei cuante este 1, adică nivelul la care $2^{-i}q = 1$. Rezolvând această ecuație vom găsi că $i = \log_2 q$.

Au fost descoperite numeroase îmbunătățiri ale algoritmului de bază, care sunt discutate în detaliu de Bertsekas și Gallager (1992). De exemplu, să considerăm cazul în care stațiile G și H vor să transmită. La nodul 1 se va produce o coliziune, așa că va fi încercat 2, care va fi găsit liber. Este fără sens să încercăm nodul 3 pentru că este sigur că vom avea o coliziune (știm că două sau mai multe stații de sub 1 vor să transmită și nici una dintre ele nu se află sub 2, deci toate sunt sub 3). Încercarea lui 3 poate fi sărită și se trece la 6. Dacă nici această încercare nu dă nici un rezultat, 7 poate fi sărit și este încercat G în continuare.

4.2.5 Protocoale cu acces multiplu cu divizarea frecvenței

O abordare diferită a problemei alocării canalului o reprezintă împărțirea acestuia în subcanale utilizând FDM, TDM, sau amândouă, și alocarea lor dinamică după necesități. Astfel de metode sunt frecvent utilizate în LAN-urile cu fibră optică pentru a permite ca transmisiuni diferite să utilizeze lungimi de undă (adică frecvențe) diferite în același timp. În această secțiune vom examina un astfel de protocol (Humblet ș.a., 1992).

O cale simplă de construire a unui LAN cu fibră optică este utilizarea unui cuplor pasiv de tip stea (vezi fig. 2-10). Două fibre de la fiecare stație intră într-un cilindru de sticlă. O fibră este pentru transmisia către cilindru iar cealaltă pentru transmisia de la cilindru. Emisia de lumină de la oricare din stații iluminează cilindrul și poate fi detectată de toate celelalte stații. Stelele pasive pot cupla până la sute de stații.

Pentru a permite transmisiuni multiple simultane, spectrul este divizat în canale (benzi de frecvență), ca în fig. 2-24. În acest protocol, **WDMA (Wavelength Division Multiple Access**, rom: acces multiplu cu divizarea frecvenței), fiecărei stații îi sunt asociate două canale. Un canal îngust este folosit drept canal de control pentru semnalizarea către stație, iar unul larg pentru ca stația să poată trimite cadre de date prin el.

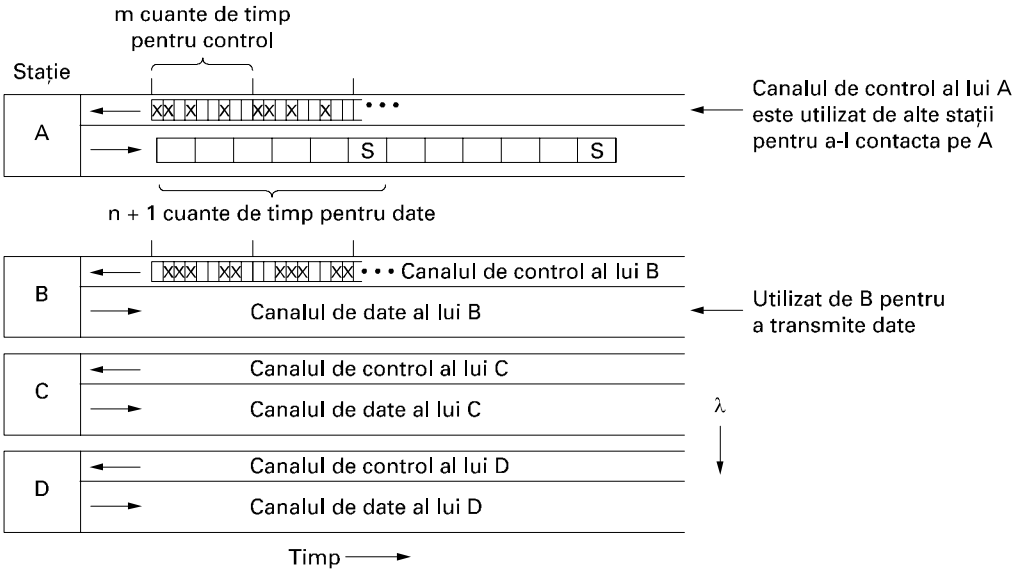


Fig. 4-10. Acces multiplu cu divizarea frecvenței.

Fiecare canal este împărțit în cuante de timp, ca în fig. 4-10. Fie m numărul de cuante ale canalului de control și $n + 1$ numărul de cuante ale canalului de date, dintre care n sunt pentru date și ultima este utilizată de stație pentru a-și raporta starea (în principal, care dintre cuantele ambelor canale sunt libere). Pe ambele canale secvența de cuante se repetă la infinit, cu cuanta 0 marcată special pentru ca cei care iau parte mai târziu la transmisie să o poată detecta. Toate canalele sunt sincronizate de un unic ceas global.

Protocolul suportă trei clase de trafic: (1) trafic orientat pe conexiune, cu rată constantă de date (cum este semnalul video necomprimat), (2) trafic orientat pe conexiune, cu rată variabilă de date (cum este transferul de fișiere) și (3) trafic de datagrame, cum sunt pachetele UDP. Pentru cele două protocoale orientate pe conexiune, ideea este că dacă A vrea să comunice cu B, trebuie să insereze mai întâi un cadru CONNECTION REQUEST (cerere conectare) într-o cuantă liberă de pe canalul de control al lui B. Dacă B acceptă, comunicația se poate desfășura pe canalul de date al lui A.

Fiecare stație are doi emițători și doi receptori, după cum urmează:

1. Un receptor cu lungime de undă fixă pentru ascultarea propriului canal de control.
2. Un emițător reglabil pentru comunicarea pe canalul de control al altei stații.
3. Un emițător cu lungime de undă fixă pentru emisia cadrelor de date.
4. Un receptor reglabil pentru selectarea emițătorului de ascultat.

Cu alte cuvinte, fiecare stație își ascultă propriul canal de control pentru cererile care sosesc, dar trebuie să se regleze pe frecvența emițătorului pentru a primi datele. Reglarea frecvenței este realizată cu un interferometru Fabry-Perot sau Mach-Zehnder, care elimină prin filtrare toate frecvențele, cu excepția celei dorite.

Să urmărim acum modul în care stația A stabilește un canal de comunicație de clasă 2 cu stația B pentru, să zicem, un transfer de fișiere. Mai întâi, A își reglează receptorul de date pe frecvența canalului de date al lui B și așteaptă cuanta de stare. Această cuantă precizează care cuante de control

sunt ocupate și care sunt libere. De exemplu, în fig. 4-10 se observă că din cele opt cuante de control ale lui *B*, 0, 4 și 5 sunt libere. Restul sunt ocupate (fapt indicat prin cruciulițe).

A își alege una din cele trei cuante de control, să zicem 4, și își inserează mesajul CONNECT REQUEST în ea. Cum *B* își ascultă permanent canalul de control, vede cererea și o aprobă acordând cuanta 4 lui *A*. Această decizie este anunțată în cuanta de stare a canalului de control. Atunci când *A* vede anunțul, va ști că s-a stabilit o conexiune unidirecțională. Dacă *A* cerea o conexiune bidirecțională, *B* ar fi trebuit să repete același algoritm cu *A*.

Este posibil ca în timp ce *A* căuta să ocupe cuanta 4 de control a lui *B*, *C* să facă același lucru. Nici o stație nu o va obține și amândouă vor observa eșecul urmărind cuanta de stare din canalul de date al lui *B*. Ele vor aștepta în continuare un interval de timp aleatoriu, după care vor încerca din nou.

În acest moment, fiecare stație are o cale fără conflicte pentru trimiterea de scurte mesaje de control către cealaltă. Pentru a realiza transferul de fișiere, *A* va trimite către *B* un mesaj de control, spunând, de exemplu, „Te rog uită-te la următoarea cuantă 3 cu date de ieșire de la mine. În ea se află un cadru de date pentru tine". Când *B* primește mesajul de control, își va regla receptorul pe canalul de ieșire al lui *A* pentru a citi cadrul de date. Bazându-se pe un protocol de nivel mai înalt, *B* poate utiliza același mecanism pentru a trimite înapoi o confirmare, dacă dorește.

De notat că apare o problemă când *A* și *C* au conexiuni către *B* și fiecare îi spune să se uite la cuanta 3. *B* va alege una dintre ele la întâmplare, iar cealaltă transmisie va fi pierdută.

La trafic constant este utilizată o variantă a acestui protocol. Atunci când *A* cere o conexiune, ea spune în același timp ceva de genul: este în regulă dacă îți voi trimite câte un cadru în fiecare cuantă 3? Dacă *B* poate accepta (adică nu și-a luat nici un angajament pentru cuanta 3), este stabilită o conexiune cu lărgime de bandă garantată. Dacă nu, *A* poate încerca din nou cu o altă propunere, în funcție de cuantele de ieșire libere.

Traficul de clasă 3 (datagrame) utilizează o altă variantă. În loc să scrie un mesaj CONNECTION REQUEST în cuanta de control pe care tocmai a găsit-o (4), va scrie un mesaj DATA FOR YOU ÎN SLOT 3 (în cuanta 3 se află date pentru tine). Dacă *B* este liberă în timpul următoarei cuante 3 de date, transmisiunea va reuși. Altfel, cadrul de date se va pierde. În acest fel nu vom avea niciodată nevoie de conexiuni.

Sunt posibile mai multe variante ale întregului protocol. De exemplu, în loc să îi asigurăm fiecărei stații propriul canal de control, toate stațiile pot partaja un singur canal de control. Fiecărei stații îi este asociat un bloc de cuante în fiecare grup, multiplexând astfel mai multe canale virtuale într-un singur canal fizic.

De asemenea ne putem descurca cu un singur emițător reglabil și un singur receptor reglabil pe stație, divizând canalul fiecărei stații în m cuante de control, urmate de $n + 1$ cuante de date. Dezavantajul constă în faptul că emițătorii trebuie să aștepte mai mult pentru a obține o cuantă de control, iar cadrele de date consecutive vor fi separate din cauza informațiilor de control de pe canal.

Au fost propuse numeroase alte protocoale WDMA, care se deosebesc prin detalii. Unele au un singur canal de control, altele au mai multe. Unele iau în considerare întârzierea de propagare, altele nu; unele consideră timpul de reglare a frecvenței ca făcând parte explicit din model, altele îl ignoră. De asemenea protocoalele se deosebesc prin complexitatea prelucrării, productivitate și scalabilitate. Când sunt folosite un număr ridicat de frecvențe sistemul poate fi numit **DWDM (Dense Wavelength Division Multiplexing** - acces multiplu dens cu divizarea frecvenței). Pentru mai multe informații, vezi (Bogineni ș.a., 1993; Chen, 1994; Goralski, 2001; Kartopoulos, 1999; Levine și Akyildiz, 1995).

4.2.6 Protocoale pentru rețele LAN fără fir

Pe măsură ce numărul de echipamente de calcul și comunicație crește, același lucru se întâmplă și cu nevoia lor de conectare la lumea exterioară. Chiar și primele telefoane portabile aveau posibilitatea de a se conecta la alte telefoane. Primele calculatoare portabile nu au avut această posibilitate, dar curând după aceea, modemurile au devenit un lucru obișnuit. Pentru a comunica, aceste calculatoare trebuiau să fie conectate la o priză telefonică de perete. Necesitatea unei conexiuni prin cablu la o rețea fixă însemna că de fapt calculatoarele, deși erau portabile, nu erau mobile.

Pentru a obține o adevărată mobilitate, calculatoarele portabile trebuie să utilizeze pentru comunicație semnale radio (sau infraroșii). Astfel, utilizatorii dedicați pot citi sau trimite poșta electronică în timp ce merg cu mașina sau cu vaporul. Un sistem de calculatoare portabile care comunică prin radio poate fi privit ca un LAN fără fir. Aceste LAN-uri au proprietăți oarecum diferite față de LAN-urile convenționale și necesită protocoale speciale pentru subnivelul MAC. În această secțiune vom examina câteva din aceste protocoale. Mai multe informații despre rețelele locale fără fir pot fi găsite în (Geier, 2002; O`Hara și Petrick, 1999).

O configurație obișnuită pentru un LAN fără fir este o clădire cu birouri, cu stațiile de bază amplasate strategic în jurul clădirii. Toate stațiile de bază sunt interconectate prin cabluri de cupru sau fibră optică. Dacă puterea de emisie a stațiilor de bază și a calculatoarelor portabile este reglată la o rază de acțiune de 3 sau 4 metri, atunci fiecare cameră devine o singură celulă, iar întreaga clădire devine un mare sistem celular, ca în sistemele de telefonie celulară tradițională, pe care le-am studiat în Cap. 2. Însă, spre deosebire de sistemele de telefonie celulară, fiecare celulă are un singur canal, acoperind întreaga lărgime de bandă disponibilă și acoperind toate stațiile din respectiva celulă. În mod normal, lărgimea de bandă a canalului este de 11-54 Mbps.

În discuția care urmează vom presupune, pentru simplificare, că toți emițătorii radio au un domeniu fix. Atunci când un receptor se află în raza a doi emițători activi, semnalul rezultat va fi, în general, amestecat și neutilizabil (cu câteva excepții care vor fi discutate mai târziu). E important să ne dăm seama că în unele LAN-uri fără fir nu toate stațiile se află în același domeniu, ceea ce duce la o serie de complicații. Mai mult, pentru LAN-uri de incintă fără fir, prezența pereților între stații poate avea un impact major asupra domeniului efectiv al fiecărei stații.

O abordare naivă în construirea unui LAN fără fir o constituie încercarea de utilizare a CSMA, prin ascultarea celorlalte transmisiuni și transmisia numai în cazul în care nimeni nu transmite. Problema este că acest protocol nu este chiar potrivit, pentru că ceea ce contează este interferența la receptor, nu la emițător. Pentru a vedea natura problemei, să privim fig. 4-11, în care apar patru stații nelegate prin cablu. Pentru ceea ce vrem să arătăm nu contează care sunt stații de bază și care sunt calculatoare portabile. Domeniul (de recepție) radio are proprietatea că *A* și *B* sunt fiecare în domeniul celeilalte și pot interfera una cu cealaltă. Și *C* poate să interfereze atât cu *B* cât și cu *D*, dar nu cu *A*.

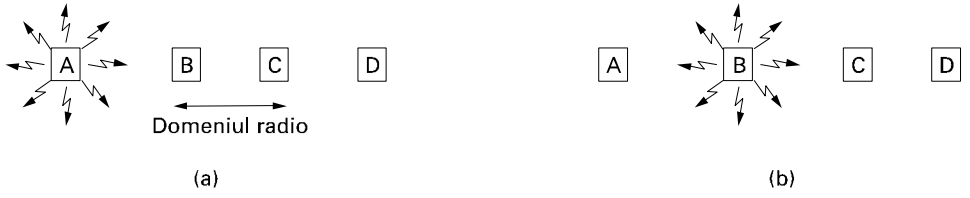


Fig. 4-11. Un LAN fără fir. (a) A transmite. (b) B transmite.

Să considerăm mai întâi ce se întâmplă atunci când *A* transmite către *B*, ca în fig. 4-11(a). Dacă *C* ascultă mediul, ea nu o va auzi pe *A* pentru că *A* este în afara domeniului ei, trăgând concluzia falsă că poate transmite. Dacă *C* începe să transmită, ea va interfera la *B* cu cadrul de la *A*, distrugându-l. Problema stației care nu poate detecta un potențial competitor la mediu pentru că se află prea departe este numită uneori **problema stației ascunse** (**hidden station problem**).

Să considerăm acum situația inversă: *B* transmite către *A*, ca în fig. 4-11(b). Dacă *C* ascultă mediul, va sesiza transmisia și va deduce în mod incorect că nu poate transmite către *D*, când de fapt o asemenea transmisie ar cauza o proastă recepție doar în zona cuprinsă între *B* și *C*, unde nu se află nici unul dintre receptorii vizați. Această situație se mai numește și **problema stației expuse** (**exposed station problem**).

Problema este că înainte de a începe o transmisiune, o stație dorește să știe dacă în preajma receptorului se desfășoară sau nu vreo activitate. CSMA sesizează acest lucru prin simpla detecție a purtătoarei. Prin cablu, toate semnalele se propagă la toate stațiile, așa că, la un moment dat, poate avea loc o singură transmisie, indiferent de zona sistemului în care se desfășoară ea. Într-un sistem bazat pe unde radio cu domeniu mic, se pot desfășura mai multe transmisiuni simultan, dacă acestea au destinații diferite și aceste destinații au domenii disjuncte.

Altă cale de abordare a acestei probleme este să ne închipuim o clădire de birouri în care fiecare angajat are un calculator portabil nelegat prin cablu. Să presupunem că Linda vrea să îi transmită un mesaj lui Milton. Calculatorul Lindei ascultă mediul local și, nedetectând nici o activitate, începe să transmită. Totuși, se mai poate produce o coliziune în biroul lui Milton, pentru că o a treia persoană îi transmitea deja dintr-un alt loc, atât de departe de Linda, încât calculatorul ei nu a putut detecta acest lucru.

MACA și MACAW

Unul dintre primele protocoale concepute pentru LAN-uri fără fir este **MACA** (**Multiple Access with Collision Avoidance** - acces multiplu cu evitarea coliziunii) (Karn, 1990). El a fost utilizat ca bază pentru standardul de LAN fără fir IEEE 802.11. Ideea de bază care stă în spatele său este ca emițătorul să stimuleze receptorul să emită un scurt cadru, astfel încât stațiile apropiate să poată detecta această transmisiune și să nu emită și ele pe durata cadrului (mare) de date care urmează. MACA este ilustrat în fig. 4-12.

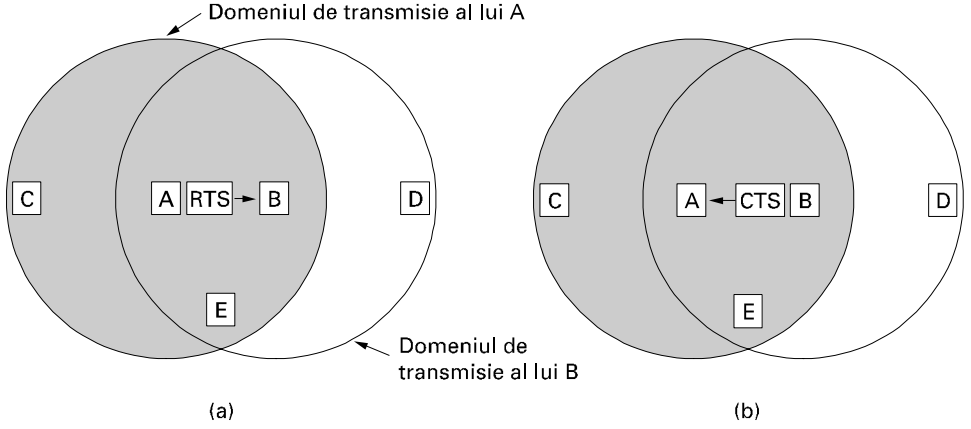


Fig. 4-12. Protocolul MACA. (a) *A* emite un RTS către B.
(b) *B* îi răspunde lui A cu un CTS.

Să vedem acum modul în care *A* îi trimite un cadru lui *B*. *A* începe prin a emite un cadru RTS (Request To Send, rom: cerere de emisie) către *B*, ca în fig. 4-12(a). Acest scurt cadru (30 de octeți) conține lungimea cadrului de date care va urma. Apoi *B* răspunde cu un cadru CTS (Clear To Send, rom: aprobare transmisie), ca în fig. 4-12(b). Cadrul CTS conține lungimea datelor (copiată din cadrul RTS). La recepția cadrului CTS, *A* începe transmisia.

Să urmărim acum modul în care reacționează stațiile care recepționează vreunul din aceste cadre. Orice stație care aude RTS se află în mod cert în apropierea lui *A* și trebuie să tacă suficient de mult timp pentru ca să poată fi trimis un CTS înapoi la *A*, fără conflicte. Orice stație care recepționează CTS se află în mod cert în apropiere de *B* și trebuie să tacă în timpul transmisiei de date în curs, a cărei lungime o poate afla examinând cadrul CTS.

În fig. 4-12, *C* se află în domeniul lui *A*, însă nu în domeniul lui *B*. De aceea va auzi RTS de la *A*, dar nu și CTS de la *B*. Cât timp nu interferează cu CTS, ea este liberă să transmită în timp ce cadrul de date este emis. În schimb *D* este în domeniul lui *B*, dar nu și în cel al lui *A*. Nu aude RTS, dar aude CTS. Recepționând CTS, își va da seama că este aproape de o stație care este pe cale să primească un cadru, așa că se va abține de la a emite ceva până când, după calculele sale, acel cadru se va termina. Stația *E* aude ambele mesaje de control și, ca și *D*, trebuie să tacă până la terminarea cadrului de date.

În ciuda acestor precauții, încă mai pot apărea coliziuni. De exemplu, *B* și *C* ar putea transmite simultan cadre RTS către *A*. Ele vor intra în coliziune și se vor pierde. În eventualitatea unei coliziuni, un emițător care nu a avut succes (adică unul care nu aude un CTS în intervalul de timp prevăzut) va aștepta o perioadă de timp aleatorie și va încerca din nou. Algoritmul utilizat este cel de regresie exponențială binară, pe care îl vom studia când vom ajunge la LAN-ul IEEE 802.3.

Bazat pe studii de simulare a MACA, Bharghavan ș.a. (1994) au reușit până la urmă să îmbunătățească performanțele MACA și au redenumit noul lor protocol **MACAW**. La început, ei au observat că, fără confirmări ale nivelului legătură de date, cadrele pierdute nu erau retransmise până când, mult mai târziu, nivelul transport le observa absența. Au rezolvat această problemă introducând un cadru de confirmare ACK după fiecare cadru de date transmis cu succes. Tot ei au mai observat că CSMA are o oarecare utilitate, și anume să oprească o stație de la a transmite un RTS concomitent cu o altă stație apropiată care face același lucru către aceeași destinație, așa că a fost adăugată și detecția de purtătoare. În plus, ei au mai decis să execute algoritmul de regresie separat pentru fiecare flux de date (pereche sursă-destinație), iar nu pentru fiecare stație. Această schimbare îmbunătățește echitatea protocolului. În final, pentru a îmbunătăți performanțele sistemului, s-au mai adăugat: un mecanism ce permite stațiilor să schimbe informații despre congestia rețelei și o cale de a face ca algoritmul de regresie să reacționeze mai puțin violent la problemele temporare.

4.3 ETHERNET

Am terminat acum discuția noastră generală despre protocoalele de alocare a canalelor în teorie, deci este timpul să vedem cum se aplică aceste principii sistemelor reale – în particular, LAN-urilor. După cum am discutat în secțiunea 1.5.3, IEEE a standardizat un număr de rețele locale și metropolitane sub numele de IEEE 802. Câteva au supraviețuit, dar nu multe, după cum am văzut în fig. 1-38. Unii dintre cei care cred în reîncarnare se gândesc că Charles Darwin s-a întors ca membru al IEEE Standards Association pentru a elimina rețelele neadaptate. Cei mai importanți dintre supra-

viețuitori sunt: 802.3 (Ethernet) și 802.11 (LAN fără fir). În ceea ce privește 802.15 (Bluetooth) și 802.16 (MAN fără fir), este prea devreme pentru a ne pronunța. Vă sfătuim să consultați ediția a 5-a a acestei cărți ca să aflați. Atât 802.3 și 802.11 au niveluri fizice diferite și subniveluri MAC diferite, dar ele converg asupra aceluiași subnivel logic de control al conexiunii (LLC) (definit în 802.2), astfel încât au aceeași interfață cu nivelul de rețea.

Am introdus Ethernetul în Sec.1.5.3 și nu vom mai repeta aici aceleași informații. În continuare ne vom concentra asupra detaliilor tehnice ale Ethernetului, protocoalele și realizările recente în Ethernet-ul de mare viteză (gigabit). Din moment ce Ethernet și IEEE 802.3 sunt aproape identice, cu excepția a două detalii minore pe care le vom discuta în curând, mulți oameni folosesc termenii „Ethernet" și „IEEE 802.3" ca sinonime, astfel încât și noi vom face același lucru. Pentru mai multe informații despre Ethernet, vezi (Bradley și Riley, 1999; Seifert, 1998; Spurgeon, 2000).

4.3.1 Cablarea Ethernet

Întrucât numele „Ethernet" se referă la cablu (eterul), să pornim discuția noastră de aici. În mod obișnuit, sunt utilizate patru tipuri de cabluri, după cum se arată în fig. 4-13.

Nume	Cablu	Seg. maxim	Noduri / seg.	Avantaje
10Base5	coaxial gros	500 m	100	Cablul original, în prezent ieșit din uz
10Base2	coaxial subțire	185 m	30	Nu este nevoie de hub
10Base-T	perechi torsadate	100 m	1024	Cel mai ieftin sistem
10Base-F	Fibră optică	2000 m	1024	Cel mai bun între clădiri

Fig. 4-13. Cele mai obișnuite tipuri de cablare Ethernet.

Din punct de vedere istoric, cablul **10Base5**, numit popular și **Ethernet gros** (**thick Ethernet**), a fost primul. El se aseamănă cu un furtun galben de grădină cu semne la fiecare 2.5 metri pentru a arăta unde vin conectorii (Standardul 802.3 nu *impune* de fapt cabluri de culoare galbenă, dar *sugerează* acest lucru). Conexiunile cu el sunt făcute în general utilizând **conectori-vampir** (**vampire taps**), la care un pin este introdus cu *mare* grijă până în miezul cablului coaxial. Notația 10Base5 înseamnă că funcționează la 10 Mbps, utilizează semnalizare în banda de bază și poate suporta segmente de până la 500 metri. Primul număr reprezintă viteza în Mbps. Apoi urmează cuvântul „Base" (uneori „BASE") pentru a indica transmisia în banda de bază. Exista mai demult o variantă în banda largă, 10Broad36, dar nu s-a impus pe piață și a dispărut. În fine, dacă mediul de transmisie este cablul coaxial, lungimea sa apare rotunjită în unități de 100m după „Base".

Istoric vorbind, al doilea tip de cablu a fost **10Base2**, sau **Ethernet subțire** (**thin Ethernet**), care, spre deosebire de Ethernet gros „ca un furtun de grădină", se îndoaie ușor. Conexiunile cu el sunt făcute utilizând conectori standard industriali BNC pentru a forma joncțiuni în T, mai curând decât conectori-vampir. Aceștia sunt mai ușor de folosit și mai siguri. Ethernetul subțire este mult mai ieftin și mai ușor de instalat, dar el poate suporta lungimi ale cablului de maxim 185 de metri pe segment, fiecare segment putând trata numai 30 de calculatoare.

Detectarea întreruperilor de cablu, a conectorilor proști sau a conectorilor desprinși poate fi o problemă majoră pentru ambele medii de transmisie. Din acest motiv au fost dezvoltate tehnici care să le detecteze. În esență, în cablu este injectat un impuls cu o formă cunoscută. Dacă impulsul întâlnește un obstacol sau ajunge la capătul cablului, va fi generat un ecou care este trimis înapoi. Măsurând cu grijă timpul scurs între emiterea impulsului și recepționarea ecoului, este posibilă

localizarea originii ecoului. Această tehnică este numită **reflectometrie în domeniul timp** (**time domain reflectometry**).

Problemele asociate cu găsirea întreruperilor de cablu au condus sistemele către un alt tip de model de cablare, în care toate stațiile au un cablu care duce la un **concentrator** (**hub**). De obicei, aceste fire sunt perechi torsadate ale companiei de telefoane, deoarece majoritatea clădirilor cu birouri sunt deja cablate în acest fel și, în mod normal, există o mulțime de perechi disponibile. Această strategie se numește **10Base-T**. Concentratorii nu pot ține într-o memorie tampon traficul pe care îl transferă. Vom discuta mai târziu în acest capitol o versiune îmbunătățită a acestei idei (comutatoarele), care au mecanisme de păstrare a traficului primit într-o memorie tampon.

Aceste trei strategii de cablare sunt ilustrate în fig. 4-14. Pentru 10Base5, în jurul cablului este prins strâns un **transiver** (**transceiver**), astfel încât conectorul său face contact cu miezul cablului. Transiverul conține partea de electronică care se ocupă cu detecția purtătoarei și cu detecția coliziunilor. Atunci când este detectată o coliziune, transiverul trimite pe cablu un semnal nepermis special, pentru a se asigura că și celelalte transivere își dau seama că s-a produs o coliziune.

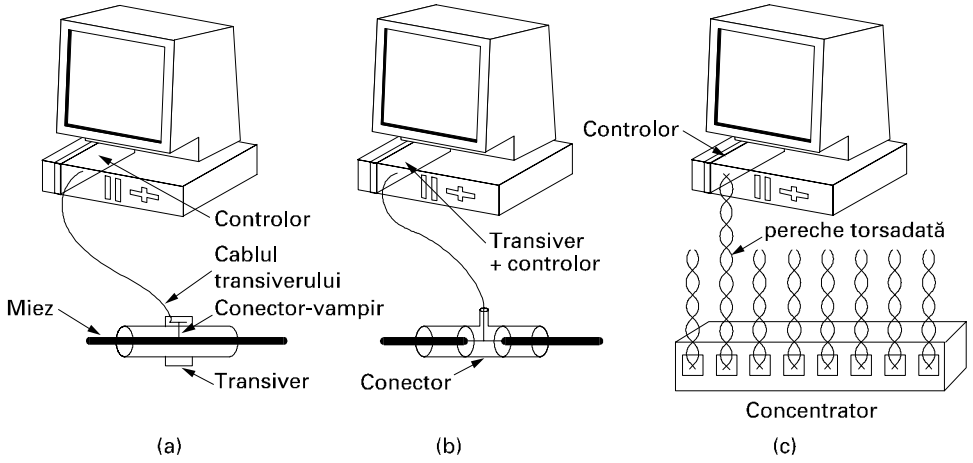


Fig. 4-14. Trei tipuri de cablare 802.3. (a) 10Base5. (b) 10Base2. (c)10Base-T.

La 10Base5, un **cablu de transiver** (**transceiver cable**) conectează transiverul cu o placă de interfață din calculator. Cablul transiverului poate avea până la 50 de metri lungime și conține cinci perechi torsadate izolate individual. Două dintre perechi sunt pentru datele de intrare și respectiv datele de ieșire. Alte două sunt pentru semnalele de control de intrare și de ieșire. A cincea pereche, care nu este întotdeauna folosită, permite calculatorului să alimenteze electronica transiverului. Pentru a reduce numărul de transivere necesare, unele transivere permit să le fie atașate până la opt calculatoare învecinate.

Cablul transiverului se termină la placa de interfață din interiorul calculatorului. Placa de interfață conține un cip controlor care transmite cadre către transiver și recepționează cadre de la acesta. Controlorul este responsabil cu asamblarea datelor în formatul de cadru corespunzător, precum și cu calculul sumelor de control pentru cadrele trimise și verificarea lor pentru cadrele primite. Unele cipuri controlor gestionează și un set de zone tampon pentru cadrele primite, o coadă de zone tampon pentru transmisie, transferurile DMA cu calculatoarele gazdă și alte aspecte legate de administrarea rețelei.

La 10Base2, conexiunea cu cablul se face printr-un conector BNC pasiv cu joncțiune în T. Electronica transiverului este pe placa controlorului și fiecare stație are întotdeauna propriul transiver.

La 10Base-T, nu există nici un cablu, ci doar un concentrator - o cutie plină de electronică. Adăugarea sau îndepărtarea unei stații este mai simplă în această configurație, iar întreruperile cablului pot fi detectate ușor. Dezavantajul lui 10base-T este acela că dimensiunea maximă a cablului care pleacă de la concentrator este de numai 100 de metri, poate chiar 150 de metri, dacă sunt folosite perechi torsadate de foarte bună calitate (categoria 5). De asemenea, un concentrator mare costă mii de dolari. Totuși, 10Base-T devine tot mai popular datorită ușurinței de întreținere. O versiune mai rapidă de 10Base-T (100Base-T) va fi discutată mai târziu în acest capitol.

A patra opțiune de cablare pentru 802.3 este **10Base-F**, care folosește fibre optice. Această alternativă este scumpă datorită costului conectorilor și a terminatorilor, dar are o imunitate excelentă la zgomot și este metoda care este aleasă atunci când transmisia se face între clădiri sau concentratoare aflate la distanțe mari. Sunt permise distante de kilometri. Oferă de asemenea o securitate bună, deoarece interceptarea traficului de pe o fibră de sticlă este mult mai dificil decât ascultarea traficului pe cablul de cupru.

Fig. 4-15 arată diferite moduri de cablare a unei clădiri. În fig. 4-15(a), un singur cablu este șerpuit din cameră în cameră, fiecare stație fiind conectată direct la el în punctul cel mai apropiat. În fig. 4-15(b), o coloană verticală suie de la parter până la acoperiș, cu cabluri orizontale conectate direct la ea la fiecare etaj prin amplificatoare speciale (repetoare). În unele clădiri, cablurile orizontale sunt subțiri, iar coloana este groasă. Cea mai generală topologie este cea de arbore, ca în fig. 4-15(c), deoarece o rețea cu două căi între unele perechi de stații poate suferi din cauza interferenței dintre cele două semnale.

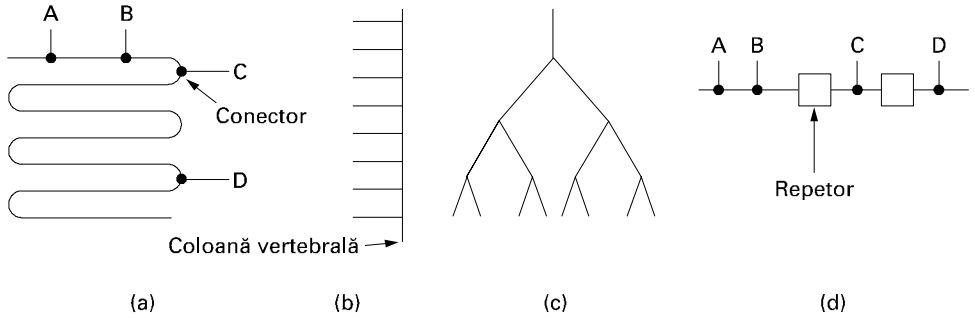


Fig. 4-15. Topologii de cablu. (a) Liniar. (b) Coloană. (c) Arbore. (d) Segmentat.

Fiecare versiune de 802.3 are o lungime maxim admisă de cablu pe segment. Pentru a permite rețele mai mari, mai multe cabluri pot fi conectate prin **repetoare** (**repeaters**), așa cum se arată în fig. 4-15(d). Un repetor este un dispozitiv de nivel fizic. El recepționează, amplifică și retransmite semnale în ambele direcții. În ceea ce privește programarea, o serie de segmente de cablu conectate prin repetoare nu prezintă nici o diferență față de un singur cablu (cu excepția unei oarecare întârzieri introduse de repetoare). Un sistem poate conține segmente de cablu multiple și repetoare multiple, dar două transivere nu pot fi la o distanță mai mare de 2,5 km și nici o cale între oricare două transivere nu poate traversa mai mult de 4 repetoare.

4.3.2 Codificarea Manchester

Nici una din versiunile lui 802.3 nu folosește o codificare binară directă, cu 0 volți pentru un bit 0 și 5 volți pentru un bit 1, deoarece aceasta conduce la ambiguități. Dacă o stație trimite șirul de biți 00010000, altele l-ar putea interpreta fals ca 10000000 sau 01000000 întrucât nu pot distinge diferența între un emițător inactiv (0 volți) și un bit 0 (0 volți). Această problemă poate fi rezolvată prin utilizarea valorilor +1V pentru 1 și –1V pentru 0. Totuși, această soluție nu rezolvă problema receptorului care va eșantiona semnalul cu o frecvență ușor diferită de cea pe care emițătorul o folosește ca să-l genereze. Ceasurile diferite pot duce la o desincronizare între emițător și receptor în ceea ce privește granițele biților, în special după un șir lung de 0 consecutivi sau de 1 consecutivi.

Ceea ce le trebuie receptorilor este un mijloc de a determina fără dubii începutul, sfârșitul și jumătatea fiecărui bit fără ajutorul unui ceas extern. Două astfel de abordări se numesc **codificarea Manchester** (**Manchester encoding**) și **codificarea Manchester diferențială** (**differential Manchester encoding**). În cazul codificării Manchester, fiecare perioadă a unui bit este împărțită în două intervale egale. Un bit 1 este trimis stabilind un voltaj ridicat în timpul primului interval și scăzut în cel de-al doilea. Un 0 binar este trimis exact invers: întâi nivelul scăzut iar apoi cel ridicat. Această strategie asigură că fiecare perioadă a unui bit are o tranziție la mijloc, ușurând sincronizarea între emițător și receptor. Un dezavantaj al codificării Manchester este acela că necesită o lărgime de bandă dublă față de codificarea binară directă, deoarece impulsurile au durata pe jumătate. Codificarea Manchester este prezentată în fig. 4-16(b).

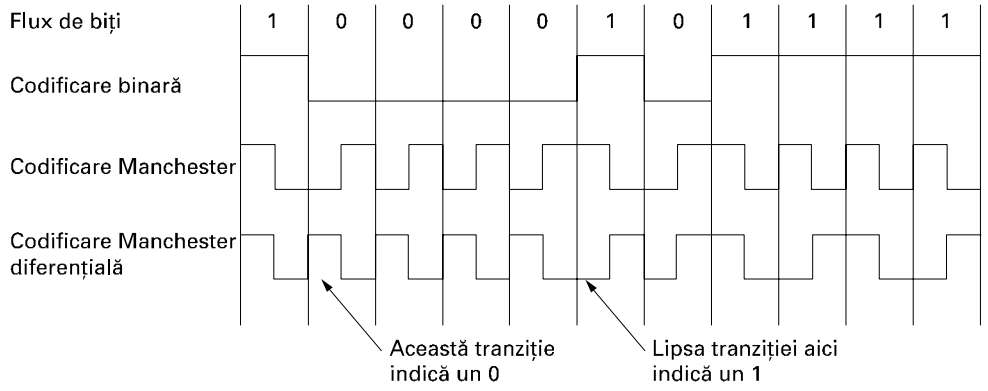


Fig. 4-16. (a) Codificare binară. (b) Codificare Manchester. (c) Codificare Manchester diferențială.

Codificarea Manchester diferențială, prezentată în fig. 4-16(c), este o variantă a codificării Manchester clasice. În aceasta, un bit 1 este indicat prin absența tranziției la începutul unui interval. Un bit 0 este indicat prin prezența unei tranziții la începutul intervalului. În ambele cazuri, există și o tranziție la mijloc. Strategia diferențială necesită un echipament mai complex, dar oferă o mai bună imunitate la zgomot. Toate sistemele 802.3 în banda de bază folosesc codificarea Manchester datorită simplității sale. Semnalul înalt este de +0.85 volți iar semnalul scăzut este de -0.85 volți, dând o valoare în curent continuu de 0 volți. Ethernet nu folosește codificarea Manchester diferențială, dar alte LAN-uri (de exemplu: 802.5 - LAN-urile de tip jeton pe inel) o folosesc.

4.3.3 Protocolul subnivelului MAC Ethernet

Structura cadrului original DIX (DEC, Intel, Xerox) este prezentată în fig. 4-17(a). Fiecare cadru începe cu un *Preambul* (Preamble) de 8 octeți, fiecare octet conținând șablonul de biți 10101010. Codificarea Manchester a acestui șablon furnizează o undă dreptunghiulară de 10 MHz timp de 6.4 μs pentru a permite ceasului receptorului să se sincronizeze cu cel al emițătorului. Ceasurile trebuie să rămână sincronizate pe durata cadrului, folosind codificarea Manchester pentru a detecta granițele biților.

Cadrul conține două adrese, una pentru destinație și una pentru sursă. Standardul permite adrese pe 2 și pe 6 octeți, dar parametrii definiți pentru standardul în banda de bază de 10 Mbps folosesc numai adrese pe 6 octeți. Bitul cel mai semnificativ al adresei destinație este 0 pentru adresele obișnuite și 1 pentru adresele de grup. Adresele de grup permit mai multor stații să asculte de la o singură adresă. Când un cadru este trimis la o adresă de grup, toate stațiile din grup îl recepționează. Trimiterea către un grup de stații este numită **multicast** (**trimitere multiplă**). Adresa având toți biții 1 este rezervată pentru **broadcast** (**difuzare**). Un cadru conținând numai biți de 1 în câmpul destinație este distribuit tuturor stațiilor din rețea. Diferența dintre trimitere multiplă și difuzare este suficient de importantă ca să merite a fi repetată: un cadru de trimitere multiplă este trimis unui grup de stații selectate pe Ethernet; un cadru de difuzare este trimis tuturor stațiilor de pe Ethernet. Deci, trimiterea multiplă este mai selectivă, dar implică gestiunea grupurilor. Difuzarea este mai imprecisă dar nu necesită nici un fel de gestiune de grup.

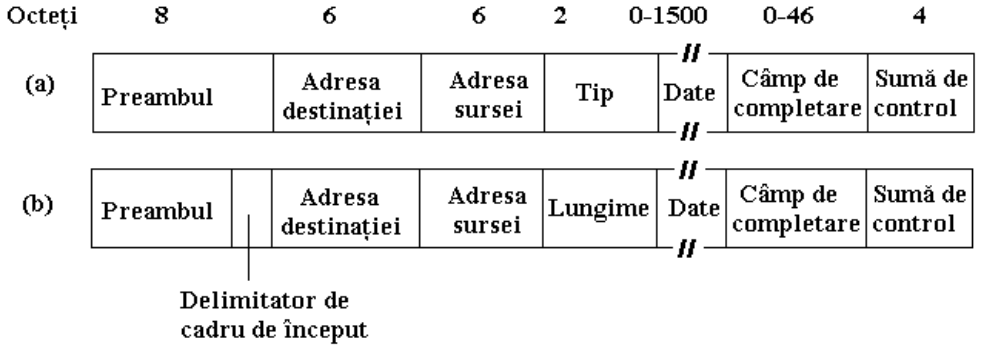


Fig. 4-17. Formatul cadrelor. (a) DIX Ethernet. (b) IEEE 802.3.

O altă trăsătură interesantă a adresării este utilizarea bitului 46 (vecin cu cel mai semnificativ bit) pentru a distinge adresele locale de cele globale. Adresele locale sunt stabilite de fiecare administrator de rețea și nu au semnificație în afara rețelei locale. În schimb, adresele globale sunt asignate de IEEE pentru a se asigura că oricare două stații din lume nu au aceeași adresă globală. Cu 48 - 2 = 46 biți disponibili, există aproximativ 7×10^{13} adrese globale. Ideea este că orice stație poate adresa în mod unic orice altă stație specificând numai numărul corect pe 48 de biți. Este sarcina nivelului rețea să-și dea seama cum să localizeze destinatarul.

În continuare urmează câmpul „Tip” (Type), care îi spune receptorului ce să facă cu cadrul. Numeroase protocoale de nivel rețea pot fi folosite simultan pe aceeași mașină, astfel încât, atunci când un cadru Ethernet ajunge, nucleul trebuie să știe cui să-i trimită cadrul. Câmpul „Tip” specifică procesul căruia îi este destinat cadrul.

Apoi urmează datele, până la 1500 de octeți. Această limită a fost aleasă oarecum arbitrar la momentul în care standardul DIX a fost solidificat, în special din cauza considerației că un transiver are nevoie de suficient RAM ca să conțină un cadru întreg și RAM era scumpă în 1978. O valoare mai mare pentru această limită ar fi însemnat mai mult RAM, deci un transiver mai scump.

În afară de faptul că există o lungime maximă a cadrelor, există și o lungime minimă a cadrelor. Deși un câmp de date de 0 octeți este uneori util, el poate duce la o situație problemă. Când un transiver detectează o coliziune, el trunchiază cadrul curent, ceea ce înseamnă că fragmente răzlețe de cadre și biți rătăciți apar mereu pe cablu. Pentru a facilita distingerea cadrelor valide de reziduuri, Ethernet cere ca toate cadrele valide să aibă cel puțin 64 de octeți, incluzând adresa destinației și suma de control. Dacă porțiunea de date dintr-un cadru este mai mică de 46 de octeți, se folosește câmpul de completare pentru a se ajunge la lungimea minimă necesară.

Un alt motiv (și mai important) de a avea o lungime minimă a cadrului este de a preveni situația în care o stație termină transmisia unui cadru scurt înainte ca primul bit să ajungă la capătul cel mai îndepărtat al cablului, unde poate intra în coliziune cu un alt cadru. Această problemă este ilustrată în fig. 4-18. La momentul 0, stația *A*, aflată la un capăt al rețelei, expediază un cadru. Să notăm cu τ timpul de propagare al cadrului până la celălalt capăt. Exact înainte de sosirea cadrului la celălalt capăt (adică la momentul $\tau - \varepsilon$), cea mai îndepărtată stație față de *A*, stația *B*, începe să transmită. Când *B* observă că primește mai multă putere decât emite, știe că a apărut o coliziune, prin urmare abandonează transmisia și generează o rafală de 48 de biți de zgomot pentru a avertiza toate celelalte stații. Aproximativ la momentul 2τ, emițătorul observă apariția zgomotului și își abandonează la rândul său transmisia. Apoi așteaptă un timp aleatoriu înainte de a încerca din nou.

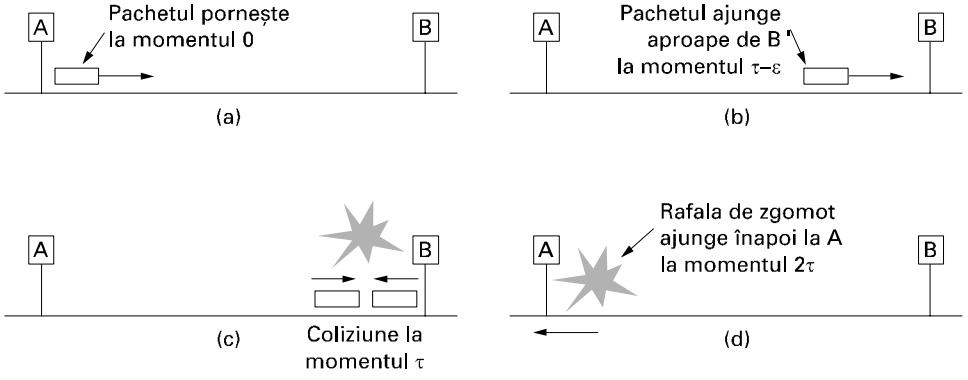


Fig. 4-18. Detectarea coliziunii poate dura 2τ.

Dacă o stație încearcă să transmită un cadru foarte scurt, este posibil să apară o coliziune, dar transmisia se termină înainte ca zgomotul produs să se întoarcă la momentul 2τ. Emițătorul va ajunge incorect la concluzia că transmisia cadrului s-a încheiat cu succes. Pentru a preveni apariția acestei situații, transmisia fiecărui cadru trebuie să ia mai mult de 2τ. Pentru un LAN la 10 Mbps cu o lungime maximă de 2500 metri și patru repetoare (conform specificației 802.3), durata unei călătorii dus-întors (incluzând și timpul necesar propagării prin cele 4 repetoare) a fost calculată la aproximativ 50 μs în cel mai defavorabil caz – inclusiv timpul trecerii prin repetoare, care în mod sigur nu este zero. Prin urmare, transmisia unui cadru minim trebuie să dureze cel puțin 50 μs pentru a se transmite. La 10 Mbps, un bit durează 100 ns, astfel încât cel mai mic cadru trebuie să aibă 500 de biți pentru o funcționare garantată. Pentru a adăuga un oarecare spațiu de siguranță, acest număr a fost

mărit la 512 biți, adică 64 de octeți. Cadrele cu mai puțin de 64 de octeți utili sunt completate până la 64 de octeți folosind câmpul de completare.

Pe măsură ce viteza rețelelor crește, lungimea minimă a cadrului trebuie să crească sau lungimea maximă a cablului trebuie să scadă proporțional. Pentru un LAN de 2500 de metri operând la 1 Gbps, dimensiunea minimă a cadrului ar trebui să fie de 6400 de octeți. Alternativ, dimensiunea minimă a cadrului ar putea fi de 640 octeți, iar distanța maximă între două stații de 250 de metri. Aceste restricții devin din ce în ce mai neplăcute pe măsură ce ne îndreptăm spre rețele cu viteze de ordinul gigabiților.

Ultimul câmp la 802.3 este *Suma de control* (*Checksum*). Aceasta este de fapt un cod de dispersie pe 32 de biți (32-bit hash-code) a datelor. Dacă anumiți biți de date sunt recepționați eronat (datorită zgomotului de pe cablu), suma de control va fi aproape sigur greșită și va fi detectată o eroare. Algoritmul sumei de control este un control cu redundanță ciclică de tipul celui discutat în cap. 3. El realizează doar detectarea erorilor și nu are legătură cu corectarea lor.

Când IEEE a standardizat Ethernetul, comitetul a decis două schimbări la formatul DIX, după cum se vede în fig. 4-17(b). Prima a fost reducerea preambulului la 7 octeți, folosind ultimul octet ca un delimitator de cadru inițial („Start of Frame") pentru compatibilizarea cu 802.4 și 802.5. A doua schimbare a constat în transformarea câmpului „tip" într-un câmp „lungime". Desigur, acum receptorul nu mai știa ce să facă cu un cadru care sosea, dar această problemă a fost rezolvată prin adăugarea unui mic antet porțiunii de date, pentru a oferi această informație. Vom discuta formatul porțiunii de date când ajungem la controlul legăturilor logice, mai târziu în acest capitol.

Din păcate, la momentul publicării lui 802.3, se utilizau deja dispozitive hardware și aplicații software pentru DIX Ethernet, astfel încât producătorii și utilizatorii nu prea erau entuziaști să convertească câmpul „tip" în câmpul „lungime". În 1997 IEEE a capitulat și a declarat că ambele standarde erau acceptabile. Din fericire, toate câmpurile „tip" folosite înainte de 1997 erau mai mari de 1500. Prin urmare, orice număr s-ar afla în acea poziție care este mai mic sau egal cu 1500 poate fi interpretat ca „lungime", iar orice număr mai mare decât 1500 poate fi interpretat ca „tip". Acum IEEE poate susține că fiecare îi folosește standardul și toată lumea poate să își vadă de treabă făcând ce făceau și înainte, fără să aibă remușcări.

4.3.4 Algoritmul de regresie exponențială binară

Să vedem acum algoritmul prin care se generează timpii aleatorii atunci când apare o coliziune. Modelul este cel din fig. 4-5. După o coliziune, timpul este împărțit în intervale discrete, a căror lungime este egală cu timpul de propagare dus-întors prin mediu în cazul cel mai defavorabil (2τ). Pentru a se potrivi cu cea mai lungă cale permisă de 802.3 (2.5 km și patru repetoare), mărimea cuantei a fost fixată la 512 intervale de bit, adică 51.2 μs – după cum a fost menționat anterior.

După prima coliziune, fiecare stație așteaptă fie 0, fie 1 cuante înainte să încerce din nou. Dacă două stații intră în coliziune și fiecare alege același număr aleatoriu, vor intra din nou în coliziune. După a doua coliziune, fiecare așteaptă la întâmplare 0, 1, 2 sau 3 cuante. Dacă se produce o a treia coliziune (probabilitatea este de 0.25), atunci, data viitoare, numărul de cuante așteptate va fi ales aleatoriu din intervalul de la 0 la $2^3 - 1$.

În general, după i coliziuni, se așteaptă un număr aleatoriu de cuante între 0 și $2^i - 1$. Oricum, după un număr de 10 coliziuni, intervalul de așteptare este înghețat la un maxim de 1023 de cuante.

După 16 coliziuni, controlorul aruncă prosopul* și raportează eșec calculatorului. Recuperarea ulterioară din situația de eroare cade în sarcina nivelurilor superioare.

Acest algoritm, numit **algoritmul de regresie exponențială binară** (**binary exponential backoff algorithm**), a fost conceput să se poată adapta dinamic la numărul stațiilor care încearcă să transmită. Dacă intervalul de generare aleatorie a fost pentru toate coliziunile 1023, șansa ca 2 stații să intre în coliziune pentru a doua oară este neglijabilă, dar timpul mediu de așteptare după o coliziune ar fi de sute de cuante, introducând o întârziere semnificativă. Pe de altă parte, dacă fiecare stație așteaptă mereu sau zero sau o cuantă, atunci dacă 100 de stații ar încerca să transmită deodată, ele ar intra în coliziune iar și iar, până când 99 dintre ele aleg 0 și una 1 sau invers. Aceasta ar putea dura ani de zile. Lăsând intervalul de generare aleatorie să crească exponențial pe măsură ce apar tot mai multe coliziuni, algoritmul asigură o întârziere minimă când se ciocnesc numai câteva stații, dar garantează de asemenea că ciocnirea este rezolvată într-un interval rezonabil atunci când este vorba de mai multe stații. Limitarea intervalului la 1023 de cuante previne creșterea peste măsura a întârzierilor.

Așa cum am arătat până acum, CSMA/CD nu oferă confirmări. Cum simpla absență a coliziunilor nu garantează că biții nu au fost modificați de zgomotul de pe cablu, pentru o comunicație sigură, destinația trebuie să verifice suma de control și, dacă este corectă, să trimită înapoi către sursă un cadru de confirmare. În mod normal, din punct de vedere al protocolului, această confirmare ar fi doar un alt cadru de date și ar trebui să lupte pentru timp de canal, ca orice cadru de date. Totuși, cu o simplă modificare a algoritmului de tratare a conflictelor s-ar permite o confirmare rapidă a recepționării cadrului (Tokoro și Tamaru, 1977): prima cuantă de conflict care urmează unei transmisii cu succes ar trebui rezervată pentru stația destinație. Din nefericire, standardul nu oferă această posibilitate.

4.3.5 Performanțele Ethernet-ului

Să examinăm pe scurt performanțele standardului 802.3 în condiții de încărcare mare și constantă, dată de k stații gata mereu să transmită. O analiză riguroasă a algoritmului de regresie exponențială binară ar fi complicată. În schimb vom proceda ca Metcalfe și Boggs (1976) și vom presupune o probabilitate de retransmisie constantă pentru fiecare cuantă. Dacă fiecare stație transmite în timpul unei cuante de conflict cu probabilitatea p, probabilitatea A ca o stație să primească canalul în această cuantă este:

$$A = kp(1 - p)^{k-1}$$

A este maxim când $p = 1/k$, și $A \rightarrow 1/e$ atunci când $k \rightarrow \infty$. Probabilitatea ca intervalul de conflict să aibă exact j cuante este $A\ (1 - A)^{j-1}$, astfel că numărul mediu de cuante pe conflict este dat de:

$$\sum_{j=0}^{\infty} jA(1 - A)^{j-1} = \frac{1}{A}$$

Întrucât fiecare cuantă durează 2τ, intervalul de conflict mediu, w, este $2\tau/A$. Presupunând p optim, numărul mediu de cuante de conflict nu este niciodată mai mare decât e, deci w este cel mult $2\tau e \approx 5.4\tau$.

* Așa procedează antrenorul unui boxer când hotărăște ca acesta să abandoneze lupta.

Dacă pentru a transmite un cadru de lungime medie sunt necesare P secunde, atunci când multe stații au cadre de transmis se obține:

$$\text{Eficiența canalului} = \frac{P}{P + 2\tau / A} \tag{4-6}$$

Aici vedem cum lungimea maximă a cablului dintre oricare două stații influențează calculul performanțelor, sugerând și alte topologii decât cea din fig. 4-15(a). Cu cât cablul este mai lung, cu atât intervalul de conflict este mai lung. Acesta este motivul pentru care standardul Ethernet specifică o lungime maximă a cablului.

Este instructiv să formulăm ecuația (4-6) și în termeni de lungime de cadru F, lărgime de bandă a rețelei B, lungime a cablului L și viteză de propagare a semnalului c, pentru cazul optim cu e cuante de conflict pe cadru. Cu $P = F/B$, ecuația (4-6) devine:

$$\text{Eficiența canalului} = \frac{1}{1 + 2BLe / cF} \tag{4-7}$$

Atunci când al doilea termen al numitorului este mare, eficiența rețelei va fi mică. Mai precis, creșterea lărgimii de bandă sau a distanței (produsul BL) reduce eficiența pentru o lungime dată a cadrului. Din nefericire, o mare parte din cercetarea în domeniul hardware-ului de rețea a țintit exact creșterea acestui produs. Oamenii doresc lărgime de bandă mare pe distanțe lungi (de exemplu, MAN-urile cu fibră optică), ceea ce sugerează că Ethernetul implementat în acest fel poate să nu fie cel mai bun sistem pentru aceste aplicații. Vom vedea alte modalități de a implementa Ethernet când ajungem la Ethernetul comutat mai târziu în acest capitol.

În fig. 4-19 este trasată eficiența canalului în funcție de numărul stațiilor gata de transmisie, pentru 2τ=51.2 μs și o rată de transmisie a datelor de 10 Mbps, folosind ecuația (4-7). Cu o mărime a cuantei de 64 de octeți, nu este surprinzător faptul că nu sunt eficiente cadrele de 64 de octeți. Pe de altă parte, cu cadre de 1024 de octeți și o valoare asimptotică de e cuante de 64 de octeți pe interval de conflict, perioada de conflict este de 174 de octeți, iar eficiența este 0.85.

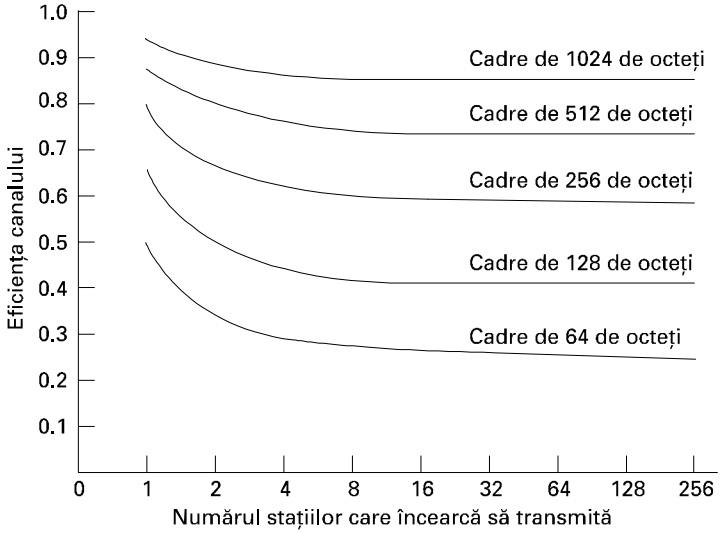


Fig. 4-19. Eficiența 802.3 la 10 Mbps cu dimensiunea cuantelor de 512 biți.

Pentru a determina numărul mediu de stații gata de transmisie în condițiile unei încărcări mari, putem să ne folosim de următoarea observație (brută). Fiecare cadru acaparează canalul pentru o perioadă de conflict și un interval de transmisie a unui cadru, totalizând un timp de $P + w$ secunde. Prin urmare, numărul de cadre pe secundă este $1/(P + w)$. Dacă fiecare stație generează cadre cu o rată medie de λ cadre/sec, atunci când sistemul este în starea k*, rata totală de intrare combinată a tuturor stațiilor neblocate este de $k\lambda$ cadre/sec. Deoarece la echilibru ratele de intrare și de ieșire trebuie să fie identice, putem egala aceste două expresii și putem rezolva pentru k (nu uitați că w este funcție de k). O analiză mai sofisticată este dată în (Bertsekas și Gallager,1992).

Probabil că merită să menționăm că s-au realizat numeroase analize teoretice ale performanțelor pentru Ethernet (și pentru alte rețele). De fapt, toată această muncă a presupus că traficul este de tip Poisson. Pe măsură ce cercetătorii au început să se uite la datele reale, s-a descoperit că traficul în rețea este rareori Poisson, în schimb este autosimilar (Paxson și Floyd,1994; și Willinger ș.a., 1995). Aceasta înseamnă că nici prin calcularea valorilor medii pe perioade lungi de timp nu se obține o netezire a traficului. Altfel spus, numărul mediu de pachete în fiecare minut al unei ore variază la fel de mult ca și numărul mediu de pachete în fiecare secundă a unui minut. Consecința acestei descoperiri este că majoritatea modelelor de trafic în rețea nu se aplică lumii reale și ar trebui luate cu un pic (sau, mai bine, cu o tonă) de sare!

4.3.6 Ethernetul comutat

Pe măsură ce la Ethernet sunt adăugate tot mai multe stații, traficul va crește. În cele din urmă, LAN-ul se va satura. O cale de ieșire din această situație este mărirea vitezei, să zicem, de la 10 Mbps la 100 Mbps. Dar, odată cu creșterea în importanță a aplicațiilor multimedia, chiar un Ethernet de 100 Mbps sau 1-Gbps poate deveni saturat.

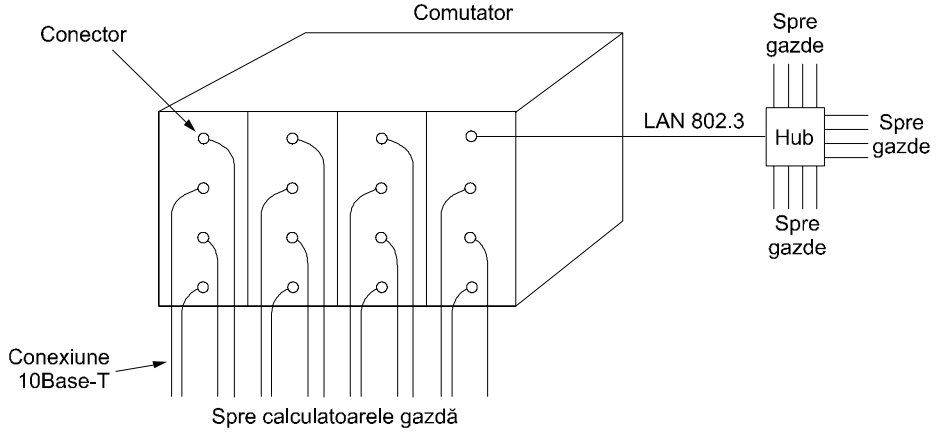


Fig. 4-20. Un LAN 802.3 comutat.

Din fericire, este posibilă o soluție diferită, mai puțin drastică: un Ethernet comutat ca cel din fig. 4-20. Inima acestui sistem este un comutator care conține o placă de bază (similară unui fund de sertar – backplane) de mare viteză și, în general, loc pentru 4 până la 32 de plăci de rețea plug-in,

* k stații gata de transmisie

fiecare având între 1 și 8 conectori. Cel mai des, fiecare conector are o conexiune prin perechi torsadate de tip 10Base-T cu un singur calculator gazdă.

Atunci când o stație dorește să transmită un cadru 802.3, trimite un cadru standard către comutator. Placa plug-in care primește cadrul verifică dacă el este destinat pentru una din celelalte stații conectate la aceeași placă. Dacă da, cadrul este copiat acolo. Dacă nu, cadrul este trimis prin placa de bază a comutatorului (backplane) către placa stației destinație. Placa de bază a comutatorului rulează în mod obișnuit la peste 1 Gbps folosind protocolul proprietar.

Ce se întâmplă dacă două calculatoare legate la aceeași placă plug-in transmit cadre în același timp? Depinde de cum a fost construită placa. O posibilitate este ca toate porturile de pe placă să fie legate împreună pentru a forma un LAN local pe placă. Coliziunile din acest LAN pe placă vor fi detectate și tratate la fel ca orice altă coliziune dintr-o rețea CSMA/CD - cu retransmisii utilizând algoritmul de regresie binară. Cu acest tip de placă plug-in este posibilă o singură transmisie pe placă la un moment dat, dar toate plăcile pot transmite în paralel. Astfel concepute, fiecare dintre plăci își formează propriul **domeniu de coliziune** (**collision domain**), independent de celelalte.

La celălalt tip de placă plug-in, fiecare port de intrare utilizează un registru tampon, astfel încât cadrele care vin sunt stocate în memoria RAM inclusă în placă, pe măsură ce sosesc. Această concepție permite tuturor porturilor de intrare să recepționeze (și să transmită) cadre în același timp, pentru operare duplex integral (full duplex), în paralel. Odată ce un cadru a fost recepționat în întregime, placa poate verifica dacă el este destinat pentru un alt port de pe aceeași placă, sau pentru un port aflat la distanță. În primul caz, el poate fi transmis direct la destinație. În cel de-al doilea, el trebuie transmis prin placa de bază a comutatorului către placa corespunzătoare. În acest mod, fiecare port este un domeniu de coliziune separat, deci nu se mai produc coliziuni. Adesea, productivitatea întregului sistem poate fi îmbunătățită astfel cu un ordin de mărime față de 10Base-5, care are un singur domeniu de coliziune pentru întreg sistemul.

Întrucât comutatorul stă și așteaptă cadre standard Ethernet pe fiecare port de intrare, putem folosi unele porturi drept concentratori. În fig. 4-20, portul din colțul din dreapta sus este conectat nu la o singură stație, ci la un concentrator cu 12 porturi. Pe măsură ce cadrele sosesc la concentrator, ele concurează pentru canale în mod obișnuit, cu apariție de coliziuni și algoritm de regresie binară. Cadrele transmise cu succes ajung la comutator, unde sunt tratate ca orice cadru de intrare: sunt îndreptate către linia de ieșire corectă prin placa de bază de viteză mare. Concentratoarele sunt mai ieftine decât comutatoarele, dar, datorită prețurilor în scădere ale comutatoarelor, ele ies treptat din uz. Totuși, mai există concentratoare rămase moștenire.

4.3.7 Ethernet-ul rapid

La început, 10 Mbps păreau raiul pe pământ, la fel cum modemurile 1200-bps păreau divine utilizatorilor modemurilor acustice de 300 bps. Totuși, noutatea s-a uzat rapid. Ca un fel de corolar al Legii lui Parkinson („Munca se dilată astfel încât să ocupe tot timpul aflat la dispoziție"), se părea că datele se dilată pentru a umple lărgimea de bandă disponibilă. Pentru a crește viteza, diverse grupuri industriale au propus două noi LAN-uri optice bazate pe două inele. Una era numită FDDI (Fiber Distributed Data Interface, rom: Interfață de Date Distribuită pe Fibră), cealaltă se numea canal de fibră (Fibre Channel[+]). Pentru a scurta o poveste lungă, amândouă au fost folosite ca și rețele de coloană vertebrală și nici una nu a reușit să ajungă în birourile utilizatorilor finali. În ambele cazuri ma-

[+] Este denumit „fibre channel" și nu „fiber channel"; autorul documentului era englez

nagementul stațiilor era prea complicat, ceea ce ducea la cip-uri complexe și prețuri ridicate. Lecția care trebuia învățată de aici era KISS (Keep it simple, Stupid, rom: Lasă lucrurile simple, prostule).

În orice caz, eșecul LAN-urilor optice în a se impune pe piață a lăsat un gol în care au înflorit o varietate de Ethernet-uri la viteze de peste 10 Mbps. Multe instalații aveau nevoie de mai multă lărgime de bandă și prin urmare aveau numeroase LAN-uri de 10 Mbps conectate printr-un labirint de repetoare, punți, rutere și porți, deși administratorilor de rețea li se părea mai degrabă că erau conectate cu gumă de mestecat și resturi de sârmă.

În acest mediu IEEE a reconvocat comitetul 802.3 în 1992 cu instrucțiuni de a produce un LAN mai rapid. O propunere a fost aceea de a păstra 802.3 exact cum era, dar să-l facă să meargă mai repede. O altă propunere era să-l refacă total astfel încât să îi ofere o mulțime de noi proprietăți, cum ar fi trafic în timp real și voce digitizată, dar să păstreze vechiul nume (din rațiuni de marketing). După ceva confruntări, comitetul a decis să păstreze 802.3 așa cum era, dar să-l facă mai rapid. Cei care susținuseră propunerea înfrântă au făcut ceea ce orice indivizi din industria de calculatoare ar fi făcut în aceste circumstanțe – s-au detașat și au format propriul lor comitet care a standardizat LAN-ul (în ceea ce va fi versiunea 802.12). Încercarea lor a eșuat lamentabil.

Comitetul 802.3 a decis să continue cu un Ethernet ameliorat din trei motive principale:

1. Nevoia de a fi compatibil retroactiv cu LAN-urile Ethernet existente;
2. Teama că un nou protocol ar putea avea consecințe negative neprevăzute;
3. Dorința de a termina treaba înainte ca tehnologia să se schimbe.

Munca a fost făcută rapid (după standardele comitetului), iar rezultatul, 802.3u, a fost aprobat oficial de IEEE în iunie 1995. Din punct de vedere tehnic, 802.3u nu este un standard nou, ci o adăugire la standardul 802.3 existent (pentru a accentua compatibilitatea cu versiunile anterioare). Din moment ce toată lumea îl denumește **Ethernet rapid** (**Fast Ethernet**), în loc de 802.3u, îl vom denumi și noi la fel.

Ideea de bază din spatele Ethernetului rapid era simplă: păstrează vechile formate de cadre, interfețele și regulile procedurale, dar reduce durata bitului de la 100 ns la 10 ns. Din punct de vedere tehnic, ar fi fost posibil să copieze fie 10Base-5 sau 10Base-2 și să detecteze în continuare coliziunile la timp pur și simplu reducând lungimea maximă a cablului cu un factor de 10. Totuși, avantajele cablării 10Base-T erau atât de copleșitoare, încât Ethernetul rapid este bazat în întregime pe acest design. Prin urmare, toate sistemele de Ethernet rapid folosesc concentratoare și comutatoare; cabluri multipunct cu conectori vampir sau BNC nu sunt permise.

Totuși, rămân câteva alegeri de făcut, dintre care cea mai importantă este ce tip de cabluri să fie suportate. Un concurent era cablul torsadat categoria 3. Argumentul pro era că practic fiecare birou în lumea occidentală are cel puțin patru cabluri răsucite categoria 3 (sau mai mult) care îl conectează cu un centru de conexiuni telefonice la cel mult 100 m distanță. Uneori există două astfel de cabluri. Prin urmare, folosind cablurile torsadate categoria 3 ar fi făcut posibilă conectarea calculatoarelor de birou la Ethernet fără să fie necesară recablarea clădirii, un avantaj enorm pentru multe organizații.

Principalul dezavantaj al cablurilor torsadate categoria 3 este incapacitatea lor de a transmite semnale de 200 megabaud (100Mbps cu codificare Manchester) pe o lungime de 100 de metri, care este distanța maximă de la calculator la concentrator specificată pentru 10Base-T (vezi fig. 4-13). Dimpotrivă, cablurile torsadate categoria 5 fac față ușor distanțelor de 100 m, iar fibra face față unor distanțe mult mai mari. Compromisul la care s-a ajuns a fost să permită toate trei posibilitățile, după cum reiese din fig. 4-21, ca să se îmbunătățească soluția de categorie 3 pentru a-i oferi capacitatea adițională de transportare de care avea nevoie.

Nume	Cablu	Segment maxim	Avantaje
100Base-T4	Cablu torsadat	100 m	Folosește UTP categoria 3
100Base-TX	Cablu torsadat	100 m	Full duplex la 100 Mbps (UTP Cat 5)
100Base-FX	Fibră de sticlă	2000 m	Full duplex la 100 Mbps; distanțe lungi

Fig. 4-21. Cablarea originală a Ethernet-ului rapid.

Schema de categorie 3 UTP, numită 100Base-T4, folosește o viteză de semnalizare de 25MHz, cu numai 25% mai rapid decât Ethernetul standard de 20MHz (amintiți-vă de codificarea Manchester care, după cum reiese din fig. 4-16, necesită două rotații de ceas pentru fiecare dintre cei 10 milioane de biți pe secundă). Totuși, pentru a obține lărgimea de bandă necesară, 100Base-T4 necesită patru perechi răsucite. Deoarece cablarea telefonică standard include de decenii patru perechi torsadate per cablu, majoritatea birourilor sunt capabile să facă față. Desigur, înseamnă să renunți la telefonul din birou, dar acesta este un preț mic pentru un e-mail mai rapid.

Din cele patru perechi torsadate una merge întotdeauna către concentrator, una vine de la concentrator, iar celelalte două sunt comutabile în direcția transmisiunii curente. Codificarea Manchester nu poate fi folosită din cauza cerințelor de lărgime de bandă, dar date fiind ceasurile moderne și distanțele scurte, nici nu mai este necesară. În plus, sunt trimise semnale ternare, astfel încât în timpul unei singure rotații de ceas cablul poate conține un 0, un 1 sau un 2. Având trei perechi torsadate în direcția „înainte" și cu semnalizare ternară, există 27 de simboluri posibile, și deci se pot trimite 4 biți cu o oarecare redundanță. Transmiterea a 4 biți în fiecare dintre cele 25 de milioane de rotații de ceas pe secundă oferă cei 100Mbps necesari. În plus, există întotdeauna un canal invers de 33.3Mbps care folosește perechea torsadată rămasă. Această schemă, cunoscută ca și 8B/6T (8 biți mapați pe 6 triți), nu este cea mai elegantă din lume, dar funcționează cu cablarea existentă.

Pentru cablarea de categorie 5, designul 100Base-TX este mai simplu deoarece cablurile fac față frecvențelor de ceas de 125MHz. Numai 2 perechi torsadate sunt folosite – una către concentrator, și alta dinspre el. Codificarea binară directă nu este folosită, ci în locul ei se află o schemă numită 4B/5B. Este preluată din FDDI și este compatibilă cu el. Fiecare grup de cinci rotații de ceas, având fiecare una dintre cele două valori ale semnalului, generează 32 de combinații. 16 dintre acestea sunt folosite pentru a transmite grupurile de biți 0000, 0001, 0010, ..., 1111. Din restul de 16, unele sunt folosite în scopuri de control, cum ar fi marcarea granițelor cadrelor. Combinațiile folosite au fost alese cu grijă, astfel încât să ofere suficiente tranziții pentru a menține sincronizarea ceasului. Sistemul 100Base-TX este integral duplex: simultan, stațiile pot transmite date la 100Mbps și pot primi date la 100Mbps. Deseori oamenii se referă la 100Base-TX și la 100Base-T4 cu denumirea comună 100Base-T.

Ultima opțiune, 100Base-Fx, folosește două linii de fibră multimod, una pentru fiecare direcție, astfel încât sistemul este, de asemenea, integral duplex, cu 100Mbps în fiecare direcție. În plus, distanța dintre o stație și concentrator poate ajunge până la 2 km.

În 1997, comitetul a adăugat, la cerere, un nou tip de cablu, 100Base-T2, permițând Ethernetului rapid să funcționeze peste două perechi de cablu de categoria 3 deja existente. Totuși, este nevoie de un procesor complicat de semnale digitale pentru a face față schemelor de codificare, așa că această opțiune este destul de scumpă. Până acum nu prea a fost utilizată, datorită complexității, costului, și faptului că multe clădiri de birouri au fost deja recablate cu categoria 5 UTP.

100Base-T face posibile două tipuri de sisteme de interconectare: concentratoare și comutatoare, după cum reiese din fig. 4-20. Într-un concentrator, toate liniile care sosesc (sau cel puțin toate liniile care ajung la o placă de extensie logică, formează un singur domeniu de coliziune. Toate regulile standard pot fi aplicate, incluzând algoritmul de regresie exponențială binară, astfel încât sistemul

funcționează exact ca Ethernetul de modă veche. În particular, o singură stație poate să transmită la un moment dat. Cu alte cuvinte, concentratoarele au nevoie de comunicații semi-duplex.

Într-un comutator, fiecare cadru care sosește este ținut într-o memorie tampon într-o placă de extensie și transmis printr-o placă de bază de mare viteză de la placa sursă la placa destinație, dacă este nevoie. Această placă de bază a comutatorului nu a fost standardizată, și nici nu trebuie să fie, din moment ce este cu desăvârșire ascunsă în interiorul comutatorului. Conform experiențelor precedente este foarte probabil că vânzătorii de comutatoare vor intra într-o concurență acerbă pentru a produce plăci de bază tot mai rapide și pentru a îmbunătăți performanța sistemului. Deoarece cablurile 100Base-FX sunt prea lungi pentru algoritmul normal de coliziune, ele trebuie să fie conectate la comutatoare, astfel încât fiecare este un domeniu de coliziune distinct. Concentratoarele nu sunt permise în 100Base-FX.

Ca observație finală, practic toate comutatoarele pot face față unui mix de stații 10 Mbps și 100 Mbps, pentru a facilita modernizarea. Pe măsură ce un site obține tot mai multe stații de 100 Mbps, tot ceea ce trebuie să facă este să cumpere numărul necesar de plăci de extensie noi și să le insereze în comutator. De fapt, standardul însuși oferă o cale astfel încât două stații să negocieze automat viteza optimă (10 sau 100Mbps) și modul de comunicație (semi-duplex sau duplex integral). Majoritatea produselor de Ethernet rapid folosesc această caracteristică pentru a se autoconfigura.

4.3.8 Ethernetul Gigabit

De-abia se uscase cerneala pe standardul Ethernetului rapid când comitetul 802 a început să lucreze la un Ethernet și mai rapid (1995). A fost numit imediat **Ethernet gigabit** (**Gigabit Ethernet**) și a fost ratificat de IEEE în 1998 sub numele 802.3z. Această notație sugerează că Ethernetul gigabit va fi sfârșitul liniei, în afară de cazul în care cineva inventează rapid o nouă literă după z. Vom discuta mai jos câteva dintre caracteristicile de bază ale Ethernetului gigabit. Mai multe informații pot fi găsite în (Seifert, 1998). Scopurile comitetului 802.3z erau practic aceleași cu ale comitetului 802.3u: să facă Ethernetul de 10 ori mai rapid, astfel încât să rămână totuși compatibil cu toate versiunile anterioare. În particular, Ethernetul gigabit trebuia să ofere suport pentru transferul fără confirmare a datagramelor atât pentru difuzare cât și pentru trimitere multiplă, să folosească aceeași schemă de adresare de 48 de biți care era deja în uz, și să mențină același format al cadrelor, inclusiv dimensiunile minime și maxime ale acestora. Standardul final a reușit să îndeplinească toate aceste scopuri.

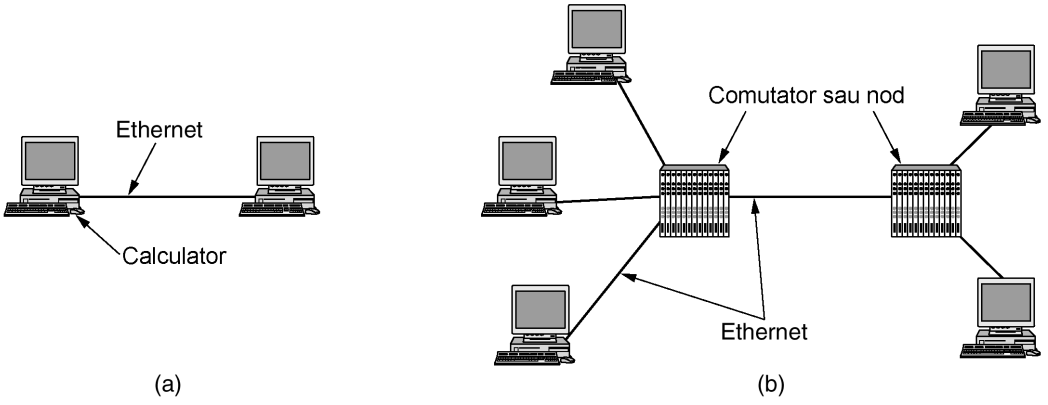


Fig. 4-22. (a) Un Ethernet cu două stații. (b) Un Ethernet cu mai multe stații.

Toate configurațiile Ethernetului gigabit sunt punct-la-punct mai degrabă decât multipunct, ca și în standardul original 10 Mbps, acum onorat cu denumirea de **Ethernet clasic**. În cea mai simpla configurație Ethernet, ilustrată în fig. 4-22(a), două calculatoare sunt conectate direct unul cu altul. Situația mai frecventă este totuși aceea în care există un concentrator sau un comutator conectat la mai multe calculatoare, și la alte concentratoare sau comutatoare adiționale, ca în fig. 4-22(b). În ambele configurații, fiecare cablu individual de Ethernet conectează exact două sisteme – nici mai multe, nici mai puține.

Ethernetul Gigabit suportă două moduri diferite de operare: modul duplex integral și modul semi-duplex. Modul „normal" este cel duplex integral, care permite traficul în ambele direcții în același timp. Acest mod este folosit atunci când există un comutator central la care sunt conectate calculatoarele (sau alte comutatoare) de la periferie. În această configurație, toate liniile sunt prevăzute cu spații tampon astfel încât fiecare calculator și fiecare comutator sunt libere să transmită cadre oricând doresc. Emițătorul nu trebuie să verifice canalul ca să vadă dacă este utilizat de altcineva, deoarece conflictele sunt imposibile. Pe linia dintre un calculator și un comutator, calculatorul este singurul emițător posibil către acel comutator și transmisia va reuși chiar și în cazul în care comutatorul transmite în același timp un cadru către calculator, deoarece linia este duplex. Din moment ce conflictele sunt imposibile, protocolul CSMA/CD nu este utilizat, astfel încât lungimea maximă a cablului este determinată de argumente referitoare la intensitatea semnalului, și nu de considerente referitoare la durata maximă a propagării zgomotului unei ciocniri către emițător. Comutatoarele sunt libere să amestece și să potrivească vitezele. Autoconfigurarea este suportată la fel ca în Ethernetul rapid.

Celălalt mod de operare, semi-duplex, este folosit când calculatoarele sunt conectate la un concentrator mai degrabă decât la un comutator. Un concentrator nu stochează cadrele care vin într-un spațiu tampon. În loc să facă asta, el conectează electric toate liniile în interior, simulând cablul multipunct folosit în Ethernetul clasic. În acest fel, există posibilitatea să apară coliziuni, astfel încât standardul CSMA/CD este necesar. Din cauză că un cadru de lungime minimă (adică de 64 de octeți) poate fi transmis acum de 100 de ori mai rapid decât în Ethernetul clasic, distanța maximă este de 100 de ori mai mică – adică de 25 de metri, pentru a menține proprietatea esențială că emițătorul mai transmite încât atunci când zgomotul ajunge înapoi la el, chiar și în cel mai rău caz. Cu un cablu lung de 2500 de metri, emițătorul unui cadru de 64 de octeți la 1Gbps va fi terminat de mult înainte ca drumul parcurs de cadru să fie măcar o zecime din cât are de mers – fără să mai socotim și returul.

Comitetul 802.3z a considerat că o rază de 25 de metri este inacceptabilă și a adăugat două caracteristici standardului pentru a mări raza. Prima caracteristică, numită **extinderea de către purtător**, se referă practic la a spune dispozitivului hardware să realinieze cadrul, mărindu-l până la 512 octeți. Din moment ce această completare este adăugată de dispozitivul hardware emițător și este înlăturată de dispozitivul hardware receptor, partea software nu este conștientă de existența sa, și prin urmare nu trebuie să sufere modificări. Desigur, transmiterea a 512 octeți de lărgime de bandă pentru a transmite 46 octeți de date ale utilizatorului (încărcătura propriu-zisă a cadrului de 64 de octeți) are o eficiență de transmitere de 9%.

A doua caracteristică, denumită **cadre în rafală (frame bursting)**, permite unui transmițător să trimită o secvență concatenată de cadre multiple într-o singură transmisie. Dacă rafala totală este mai mică de 512 octeți, dispozitivul hardware o completează din nou până la 512 octeți. Dacă sunt destule cadre care așteaptă să fie transmise, această schemă este foarte eficientă și este preferată extinderii de către purtător. Aceste noi caracteristici extind raza la 200 de metri, ceea ce probabil este suficient pentru majoritatea birourilor.

Ca să fim sinceri, este destul de greu să ne imaginăm o organizație trecând prin toate dificultățile cumpărării și instalării plăcilor de Ethernet gigabit pentru a obține o performanță ridicată, și apoi conectând calculatoarele printr-un concentrator pentru a simula Ethernetul clasic cu toate coliziunile sale. Deși concentratoarele sunt oarecum mai ieftine decât comutatoarele, plăcile de Ethernet gigabit sunt totuși scumpe. Să faci economii prin cumpărarea unui concentrator ieftin și astfel să reduci performanța noului sistem este o prostie. Totuși, compatibilitatea cu versiunile anterioare este sacră în industria calculatoarelor, astfel încât comitetul 802.3z a trebuit să se conformeze.

Ethernetul gigabit suportă atât cablarea cu cupru cât și cablarea cu fibră, precum este descris în fig. 4-23. Semnalizarea la nivelul de 1Gbps sau în jurul acestei viteze, înseamnă că sursa de lumină trebuie să fie închisă și deschisă în mai puțin de 1ns. LED-urile pur și simplu nu pot lucra atât de rapid, astfel încât este nevoie de lasere. Două lungimi de undă sunt permise: 0.85 microni (scurt) și 1.3 microni (lung). Laserele de 0.85 microni sunt mai ieftine dar nu funcționează pe fibra mono-mod.

Nume	Cablu	Segment maxim	Avantaje
1000Base-SX	Fibră de sticlă	550 m	Fibră multimod (50 și 62,5 microni)
1000Base-LX	Fibră de sticlă	5000 m	Mono-mod (10μ) sau multimod (50 și 62,5 μ)
1000Base-CX	2 perechi de STP	25 m	Pereche torsadată ecranată
1000Base-T	4 perechi de UTP	100 m	UTP Categoria 5

Fig. 4-23. Cablarea pentru Ethernet gigabit.

Sunt permise trei diametre de fibră: 10, 50 și 62,5 microni. Prima este pentru mono-mod și celelalte două sunt pentru multimod. Nu toate cele șase combinații sunt permise, totuși, iar distanța maximă depinde de combinația folosită. Numerele date în fig. 4-23 se referă la cazul cel mai fericit. În particular, 5000 de metri pot fi obișnuiți numai dacă lasere de 1,3 microni operează pe fibră de 10 microni mono-mod, dar aceasta este cea mai bună alegere pentru structurile vertebrale din campusuri și este de așteptat să fie populară, deși este și cea mai scumpă alegere.

Opțiunea 1000Base-CX folosește cabluri de cupru scurte și protejate. Problema sa este că se află în concurență cu versiunea cu fibră de înaltă performanță prezentată mai sus și cu versiunea ieftină UTP de mai jos. Este destul de puțin probabil să fie folosită la scară largă, în cele din urmă.

Ultima opțiune se referă la smocuri de patru cabluri UTP de categoria 5 lucrând împreună. Deoarece aceste cabluri sunt deja instalate în multe cazuri, este probabil că acest Ethernet gigabit va fi cel adoptat de clienții cu buzunare strâmte.

Ethernetul gigabit folosește reguli noi de codificare pe fibre. Codificarea Manchester la 1 Gbps ar ave anevoie de un semnal de 2 Gbaud, care a fost considerat foarte dificil și de asemenea foarte risipitor în ceea ce privește banda. A fost aleasă în loc o nouă schemă, numită 8B/10B, bazată pe canale de fibră. Fiecare octet de 8 biți este codificat pe fibră ca 10 biți, de unde și denumirea de 8B/10B. Din moment ce există 1024 cuvinte de cod de ieșire pentru fiecare octet de intrare, exista un oarecare spațiu de alegere în ceea ce privește cuvintele care să fie permise. Următoarele două reguli au fost folosite pentru a lua o decizie:

1. Nici un cuvânt de cod nu poate avea mai mult de patru biți identici la rând;
2. Nici un cuvânt de cod nu poate avea mai mult de șase de 0 sau șase de 1.

Aceste alegeri urmăreau să păstreze destule transmisiuni pe flux pentru a se asigura că receptorul rămâne sincronizat cu emițătorul, și de asemenea pentru a păstra numărul de 0-uri și de 1-uri pe fibră pe cât posibil egale între ele. În plus, pentru mulți octeți de intrare există două cuvinte de cod

care pot fi atribuite. Când codificatorul are de făcut o alegere, va alege întotdeauna varianta care va egaliza numărul de 0 și 1 transmiși până la momentul respectiv. Accentul este pus pe echilibrarea 0-urilor și 1-urilor pentru a păstra componenta continuă a semnalului la un nivel cât mai scăzut cu putință și pentru a-i permite să treacă nemodificată prin transformatoare. Deși cercetătorii în domeniul calculatoarelor nu sunt prea încântați de faptul că proprietățile transformatoarelor le dictează schemele de codificare, așa se întâmplă în viață uneori.

Ethernetul gigabit care folosește 1000Base-T utilizează o schemă diferită de codificare deoarece sincronizarea datelor pe un cablu de cupru într-un interval de 1ns este prea dificilă. Această soluție folosește patru cabluri torsadate de categorie 5 pentru a permite unui număr de 4 simboluri să fie transmise în paralel. Fiecare simbol este codificat folosind unul din cele cinci niveluri de voltaj. Această schemă permite ca un singur simbol să fie codificat 00, 01, 10, 11 sau cu o valoare specială în scop de control. Prin urmare, există doi biți de date per pereche torsadată, sau 8 biți de date per ciclu de ceas. Ceasul funcționează la 125 MHz, permițând operarea la 1 Gbps. Motivul pentru care sunt permise cinci niveluri de voltaj în loc de patru este necesitatea de a avea combinații rămase disponibile în scopuri de control și delimitare.

O viteză de 1 Gbps este destul de mare. De exemplu, dacă un receptor este ocupat cu o altă sarcină chiar pentru 1 ms și nu golește spațiul tampon de pe vreo linie, până atunci este posibil să se fi acumulat chiar și 1953 cadre, în acel interval de 1 ms. De asemenea, dacă un calculator care folosește Ethernet gigabit transmite date unui calculator care folosește Ethernet clasic, este foarte probabil ca memoria tampon a celui din urmă să fie epuizată, iar cadrele următoare să fie pierdute. Ca o consecință a acestor două observații, Ethernetul gigabit suportă fluxuri de control (ca și Ethernetul rapid, deși cele două sunt diferite).

Flux de control înseamnă că un capăt trimite un cadru special de control către celălalt capăt, spunându-i să ia o pauză pentru o anumită perioadă de timp. Cadrele de control sunt în general cadre Ethernet având tipul 0x8808. Primii doi octeți din câmpul de date dau comanda; următorii octeți oferă parametrii, dacă există vreunul. Pentru fluxul de control sunt folosiți cadre PAUSE, în care parametrii specifică lungimea pauzei, în unități de durată minimă a cadrului. Pentru Ethernetul gigabit unitatea de timp este de 512 ns, permițând pauze de maxim 33,6 ms.

Imediat după ce Ethernetul gigabit a fost standardizat, comitetul 802 s-a plictisit și își dorea să treacă înapoi la treabă. IEEE le-a spus să înceapă să lucreze la un Ethernet de 10-gigabit. După ce au căutat îndelung o literă care să-i urmeze lui z, au abandonat această abordare și au trecut la sufixe din două litere. S-au apucat de treabă și standardul a fost aprobat de IEEE în 2002 ca 802.3ae. Oare cât de departe poate fi Ethernetul de 100-gigabit?

4.3.9 IEEE 802.2: Controlul legăturilor logice

Acum este momentul să ne întoarcem la discuțiile anterioare și să comparăm ce am învățat în acest capitol cu ce am studiat în capitolul precedent. În cap. 3 am văzut cum două calculatoare pot comunica sigur printr-o linie nesigură folosind diferite protocoale de legături de date. Aceste protocoale ofereau controlul erorilor (prin mesaje de confirmare) precum și controlul fluxului de date (folosind o fereastră glisantă).

Dimpotrivă, în acest capitol nu am vorbit deloc despre comunicații stabile. Tot ceea ce oferă Ethernetul, ca și celelalte protocoale 802, este un serviciu datagramă de tipul „best-effort” (cea mai bună încercare). Uneori, acest serviciu este adecvat. De exemplu, în cazul transportării pachetelor

IP, nu sunt cerute și nici măcar nu sunt așteptate garanții. Un pachet IP poate să fie inserat într-un câmp de informație utilă 802 și trimis încotro o fi. Dacă se pierde, asta e.

Totuși, există și sisteme în care este de dorit un protocol de legătură de date cu control al erorilor și al fluxului. IEEE a definit un astfel de protocol care poate funcționa peste Ethernet și peste celelalte protocoale 802. Mai mult, acest protocol, numit LLC (Logical Link Control, rom: controlul legăturilor logice), ascunde diferențele între diferitele tipuri de rețele 802, oferind un singur format și o singură interfață pentru nivelul rețea. Formatul, interfața și protocolul sunt bazate îndeaproape pe protocolul HDLC, pe care l-am studiat în cap. 3. LLC formează jumătatea superioară a nivelului legătură de date, având nivelul MAC dedesubt, după cum se vede în fig. 4-24.

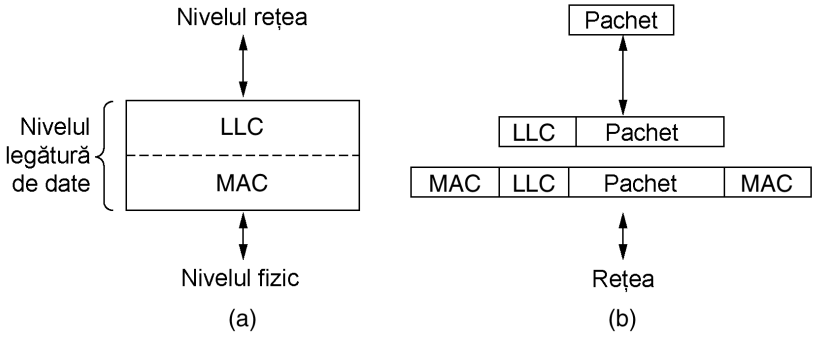


Fig. 4-24. (a) Poziția LLC. (b) Formatul protocoalelor.

Utilizarea tipică a LLC este prezentată în continuare. Nivelul rețea de pe calculatorul emițător trimite un pachet către LLC, folosind primitivele de acces LLC. Subnivelul LLC adaugă apoi un antet LLC, conținând numere care indică secvența și mesajul de confirmare. Structura rezultată este inserată apoi în câmpul de informație utilă al unui cadru 802 și apoi transmisă. Când cadrul ajunge la receptor se desfășoară procesul invers.

LLC oferă trei opțiuni de servicii: servicii pentru datagrame nesigure, confirmarea serviciului de datagrame, și un serviciu sigur orientat spre conexiuni. Antetul LLC conține trei câmpuri: un punct de acces de destinație, un punct de acces sursă și un câmp de control. Punctul de acces spune din partea cărui proces a sosit cadrul și unde trebuie transportat, înlocuind câmpul „tip” DIX. Câmpul de control conține numere de secvență și de confirmare, în stilul lui HDLC (vezi fig. 3-24), dar nu identic cu acesta. Aceste câmpuri sunt folosite în principal atunci când este necesară o conexiune stabilă la nivelul legătură de date, caz în care ar fi folosite protocoale similare cu cele discutate în cap. 3. Pentru Internet, încercările de a transmite pachete IP fără garanții sunt suficiente, astfel încât nu este nevoie de confirmări la nivelul LLC.

4.3.10 Retrospectiva Ethernetului

Ethernetul funcționează deja de 20 de ani și încă nu există competitori serioși, așa că probabil va mai funcționa încă mulți ani. Puține arhitecturi CPU, sisteme de operare sau limbaje de programare au dominat scena pentru două sau trei decenii. Evident, Ethernetul a făcut ceva cum trebuie. Ce anume?

Probabil că motivul principal al longevității sale este că Ethernetul este simplu și flexibil. Din punct de vedere practic simplu înseamnă: stabil, ieftin, și ușor de întreținut. Odată ce conectorii vampir au fost înlocuiți de conectori BNC, eșecurile au devenit extrem de rare. Oamenii ezită să înlocuiască ceva ce merge perfect tot timpul, mai ales când știu că o grămadă de lucruri din industria

calculatoarelor merg foarte prost, astfel încât multe dintre așa numitele îmbunătățiri funcționează semnificativ mai prost decât versiunea pe care au înlocuit-o.

Simplu înseamnă de asemenea ieftin. Ethernetul subțire și cablarea cu cabluri torsadate sunt relativ ieftine. Plăcile de rețea nu sunt nici ele scumpe. Doar când au fost introduse concentratoarele și comutatoarele au fost necesare investiții substanțiale, dar în momentul în care acestea au apărut în peisaj, Ethernetul era deja solid stabilit.

Ethernetul este ușor de întreținut. Nu trebuie instalat nici un software (în afara driver-elor) și nu există tabele de configurații care să trebuiască administrate (și astfel să fie un prilej de greșeli). De asemenea, adăugarea unor noi stații nu înseamnă nimic mai mult decât introducerea unui cablu în placa lor de rețea.

Un alt aspect este faptul că Ethernetul se integrează ușor cu TCP/IP, care a devenit dominant. IP este un protocol fără conexiune, ceea ce se potrivește perfect cu Ethernetul, care nici el nu este orientat pe conexiune. De exemplu, IP se potrivește mult mai greu cu ATM, care este orientat spre conexiune și această nepotrivire este un dezavantaj serios în impunerea ATM.

În cele din urmă, Ethernetul a fost capabil să evolueze în anumite aspecte cruciale. Vitezele au crescut cu câteva ordine de mărime, au fost introduse concentratoarele și comutatoarele, iar aceste schimbări nu au necesitat schimbarea interfețelor software. Dacă un vânzător din domeniul rețelelor vă arată o instalație amplă și vă spune „am această nouă rețea fantastică pentru Dvs. Tot ce trebuie să faceți este să vă aruncați tot hardware-ul și să vă rescrieți tot software-ul”, atunci are o problemă. FDDI, Canal de fibră și ATM au fost toate mai rapide decât Ethernetul când au fost introduse, dar erau incompatibile cu Ethernetul, mult mai complexe și mai dificil de administrat. În cele din urmă Ethernetul le-a ajuns din urmă în ceea ce privește viteza, astfel încât, rămase fără nici un avantaj, au murit în tăcere – cu excepția ATM care este folosit în interiorul sistemului de telefonie.

4.4 REȚELE LOCALE FĂRĂ FIR

Deși Ethernetul este folosit pe scară largă, competiția este pe cale să apară. LAN-urile fără fir sunt din ce în ce mai populare, și tot mai multe clădiri, aeroporturi și alte spații publice sunt echipate cu ele. LAN-urile fără fir pot opera în două configurații, după cum am văzut în fig. 1-35: cu sau fără stație de bază. Prin urmare, standardul LAN 802.11 ia acest fapt în considerare și oferă sprijin pentru ambele aranjamente, după cum vom vedea în continuare.

Am oferit niște informații introductive despre 802.11 în secțiunea 1.5.4. Acum este momentul să ne uităm mai îndeaproape la tehnologie. În secțiunile următoare ne vom uita la stiva de protocoale, la tehnicile de la nivelul fizic radio de transmisiuni, la protocolul subnivelului MAC, la structura cadrelor și la servicii. Pentru mai multă informație despre 802.11 vezi (Crov et. al., 1997; Geier, 2002; Heegard et. al, 2001; Kapp, 2002; O`Hara și Petrick, 1999; Severance, 1999). Pentru a afla adevărul chiar de la sursă, consultați standardul publicat al 802.11.

4.4.1. Stiva de protocoale 802.11

Protocoalele folosite de toate variantele 802, inclusiv Ethernetul, au o anumită similaritate a structurii. O viziune parțială a stivei de protocoale 802.11 este prezentată în fig. 4-25. Nivelul fizic

corespunde destul de bine cu nivelul fizic OSI, dar nivelul de legătură de date în toate protocoalele 802 este divizat în două sau mai multe subniveluri. În 802.11, subnivelul MAC (Medium Access Control) determina alocarea canalului, și anume cine va transmite următorul. Deasupra sa se află subnivelul LLC (Logical Link Control), a cărui treabă este să ascundă diferențele dintre diferitele variante 802 și să le facă să pară la fel pentru nivelul rețea. Am studiat LLC mai devreme în acest capitol, atunci când am discutat Ethernetul, astfel încât nu vom repeta această informație aici.

Standardul 802.11 din 1997 specifică trei tehnici de transmisie permise la nivelul fizic. Metoda infraroșu folosește cam aceeași tehnologie ca și telecomenzile TV. Celelalte două folosesc transmisia radio pe distanță scurtă, prin tehnici denumite FHSS și DSSS. Ambele utilizează o parte a spectrului care nu necesită licențe (banda 2,4 GHz ISM). Ușile de garaj care se deschid prin mesaje radio folosesc tot această parte a spectrului, astfel încât calculatorul vostru s-ar putea afla în competiție cu ușa de la garaj. Telefoanele fără fie și cuptoarele cu microunde folosesc și ele această bandă. Toate aceste tehnici operează la 1Mbps sau 2Mbps și cu putere suficient de mică astfel încât nu intră prea mult în conflict. În 1999 au fost introduse două noi tehnici pentru a obține o bandă mai largă. Acestea sunt denumite OFDM și HR-DSSS. Ele operează până la 54Mbps și respectiv 11Mbps. În 2001, o a doua modulație OFDM a fost introdusă, dar într-o bandă de frecvență diferită de prima. În continuare le vom examina pe fiecare pe scurt. Din punct de vedere tehnic acestea aparțin nivelului fizic și ar fi trebuit să fie examinate în cap. 2, dar deoarece sunt atât de strâns legate de rețelele locale în general și de subnivelul 802.11 MAC, le abordăm mai degrabă aici.

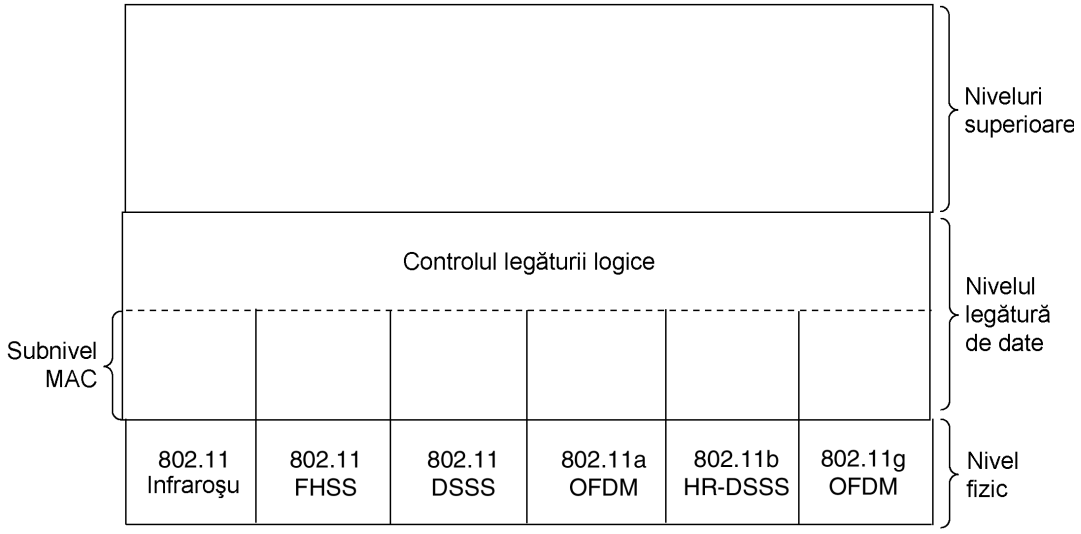


Fig. 4-25. Parte a stivei protocolului 802.11.

4.4.2. Nivelul fizic al 802.11

Fiecare dintre cele cinci tehnici de transmitere permise face posibilă trimiterea unui cadru MAC de la o stație la alta. Totuși, ele diferă în ceea ce privește tehnologia folosită și vitezele la care pot ajunge. O discuție detaliată a acestor tehnologii este cu mult în afara cadrului acestei cărți, dar câteva cuvinte despre fiecare, precum și menționarea cuvintelor cheie vor oferi cititorilor interesați termenii necesari pentru a găsi mai multă informație pe Internet sau în altă parte.

Soluția bazată pe infraroșu folosește transmisiuni cu difuzare (adică fără vizibilitate directă) la 0,85 sau 0,95 microni. Sunt permise două viteze: 1Mbps și 2Mbps. La 1Mbps se folosește o schemă de codificare în care un grup de 4 biți este codificat ca un cuvânt de 16 biți conținând 15 de 0 și un singur 1, prin ceea ce se numește **codul Gray**. Acest cod are proprietatea că o mică eroare în sincronizarea temporală duce doar la o eroare de un bit în output. La 2Mbps, codificarea ia 2 biți și produce un cuvânt codificat de 4 biți, de asemenea cu un singur 1 – adică unul dintre 0001, 0010, 0100 și 1000. Semnalele infraroșii nu pot trece prin ziduri, deci celulele din camere diferite sunt bine izolate unele de altele. Totuși, datorită lărgimii de bandă reduse (și faptului că lumina soarelui afectează semnalele în infraroșu), aceasta nu este o opțiune populară.

FHSS (**Frequency Hopping Spread Spectrum**, rom: salturi de frecvență într-un spectru larg) folosește 79 de canale, fiecare de 1MHz, începând la nivelul inferior al benzii de 2.4GHz ISM. Un generator de numere pseudo-aleator este utilizat pentru a produce secvența de frecvențe după care se vor efectua salturile. Cât timp stațiile folosesc aceeași rădăcină pentru generatorul de numere pseudo-aleatoare și stau sincronizate, ele vor sări simultan la aceleași frecvențe. Durata petrecută pe fiecare frecvență, denumită „timpul de locuire", este un parametru ajustabil, dar trebuie să fie mai mică de 400 ms. Factorul aleator al FHSS oferă o metodă eficientă de alocare a spectrului în banda ISM care nu este reglementată. De asemenea, oferă un minimum de securitate, deoarece un intrus care nu cunoaște secvența de salt sau timpul de locuire nu poate trage cu urechea la transmisiuni. Dacă distanțele sunt mari, apare problema atenuării la transmisia pe mai multe căi, dar FHSS oferă o rezistență bună. Este de asemenea relativ insensibil la interferența radio, ceea ce îl face popular pentru legăturile dintre clădiri. Principalul său dezavantaj îl constituie lărgimea de bandă redusă.

Ce-a de-a treia metodă de modulare, **DSSS** (**Direct Sequence Spread Spectrum**, rom: spectru larg cu succesiune directă) este de asemenea limitată la 1Mbps sau 2Mbps. Această schemă are anumite similarități cu sistemul CDMA pe care l-am examinat în secțiunea 2.6.2, dar diferă în alte privințe. Fiecare bit este transmis ca o secvență de 11 fragmente, folosind ceea ce se numește **secvență Barker**. Este folosită schimbarea modulării în fază la 1Mbaud, transmițând un bit per baud când operează la 1Mbps și 2 biți per baud când operează la 2Mbps. O bună bucată de timp, FCC a cerut ca toate echipamentele de comunicații fără fir care operează în banda ISM în Statele Unite să folosească împrăștierea spectrului, dar această regulă a fost abandonată în mai 2002, datorită apariției unor noi tehnologii.

Prima dintre rețelele locale fără fir de mare viteză, 802.11a, folosește **OFDM** (**Orthogonal Frequency Division Multiplexing**, rom: multiplexare cu divizare în frecvențe ortogonale) pentru a transmite până la 54Mbps în banda mai largă de 5GHz ISM. După cum sugerează și termenul FDM, sunt folosite diferite frecvențe – un număr de 52 de frecvențe, 48 pentru date și 4 pentru sincronizare – similar cu ADSL. Din moment ce transmisiunile sunt prezente pe frecvențe multiple în același timp, această tehnică este considerată o formă de împrăștiere a spectrului, dar diferită de CDMA și FHSS. Divizarea semnalului în mai multe benzi înguste are anumite avantaje comparativ cu folosirea unei benzi unice largi, incluzând o imunitate mai bună la interferența de bandă îngustă și posibilitatea utilizării benzilor care nu sunt contigue. Este folosit un sistem complex de codificare, bazat pe modularea schimbării de fază pentru viteze de până la 10 Mbps și pe QAM la viteze superioare. La 54 Mbps, 216 de biți de date sunt codificați în simboluri de 288 de biți. O parte din motivația pentru OFDM este compatibilitatea cu sistemul European HiperLAN/2 (Doufexi et al., 2002). Tehnica are o eficiență bună a spectrului în termeni de biți / Hz și o imunitate bună în fața atenuării la transmisia pe mai multe căi.

În continuare, am ajuns la **HR-DSSS** (**High Rate Direct Sequence Spread Spectrum**, rom: spectru larg cu succesiune directă la rată ridicată), o altă tehnică de spectru larg, care folosește 11 milioane de fragmente/secundă pentru a obține 11Mbps în banda de 2,4 GHz. Este denumită **802.11b**, dar nu este o continuare pentru 802.11a. De fapt, standardul său a fost aprobat primul și lansat pe piață primul. Vitezele de date suportate de 802.11b sunt 1, 2, 5,5 și 11 Mbps. Cele două viteze reduse se obțin la 1Mbaud, cu 1 și respectiv 2 biți per baud, folosind modularea schimbării de fază (pentru a fi compatibil cu DSSS). Cele două viteze mai mari se obțin la 1,375 Mbaud, cu 4 și respectiv 8 biți per baud, folosind codurile **Walsh/Haramard**. Viteza datelor poate fi adaptată dinamic în timpul operării, pentru a obține viteza optimă posibilă în condițiile curente de încărcare și zgomot. În practică, viteza operațională a lui 802.11b este de aproape 11Mbps. Deși 802.11b este mai încet decât 802.11a, spațiul său de variație este de aproape șapte ori mai mare, ceea ce este mai important în multe situații.

O versiune îmbunătățită a 802.11b, 802.11g, a fost aprobată de IEEE în noiembrie 2001, după multe dispute în legătură cu ce tehnologie patentată să folosească. Aceasta utilizează modulația OFDM a versiunii 802.11a, dar operează în banda îngustă 2,4 GHz ISM ca și 802.11b. Teoretic vorbind, poate opera la peste 54Mbps. Nu este încă foarte clar precizat momentul când această viteză va fi realizată în practică. Prin urmare, comitetul 802.11 a produs trei rețele locale fără fir: 802.11a, 802.11b și 802.11c (fără să menționăm trei rețele locale fără fir de viteză redusă). Ne-am putea pune întrebarea legitimă dacă asta este un lucru bun pentru un comitet de standardizare. Poate că trei este numărul lor cu noroc.

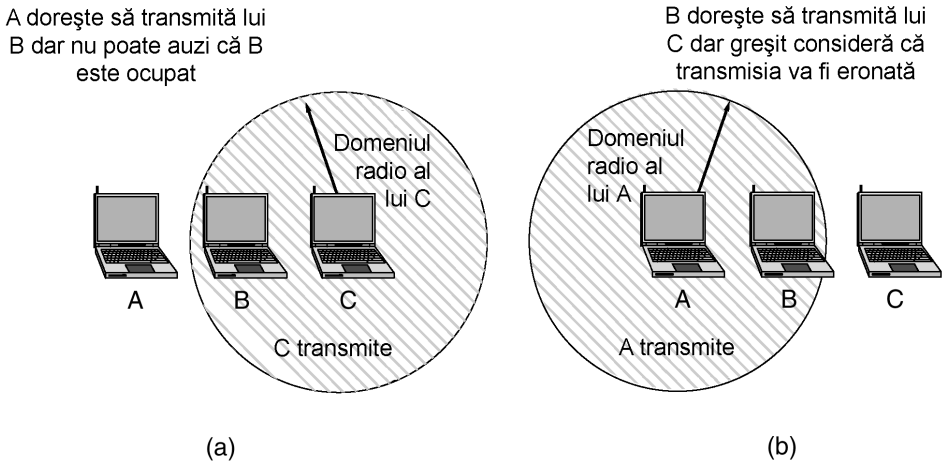


Fig. 4-26. (a) Problema stației ascunse. (b) Problema stației expuse.

4.4.3 Protocolul subnivelului MAC al 802.11

Să ne întoarcem acum din domeniul ingineriei electrice în domeniul științei calculatoarelor. Protocolul subnivelului MAC al 802.11 este destul de diferit de acela al Ethernetului datorită complexității inerente a mediului fără fir, comparativ cu un sistem de cabluri. În Ethernet o stație așteaptă până când eterul a tăcut și apoi începe să transmită. Dacă nu aude un zgomot de ciocnire în primii 64 de octeți înseamnă ca aproape sigur cadrul a fost recepționat corect. În cazul rețelelor fără fir, situația e diferită.

Pentru început, apare problema stației ascunse menționată anterior și ilustrată în fig. 4-26(a). Din moment ce nu toate stațiile se află în domeniul de acces radio una față de alta, transmisiunile

care se petrec într-o parte a celulei pot să nu fie recepționate în altă parte a celulei. În acest exemplu, stația C transmite stație B. Dacă A observă canalul, nu va auzi nimic și va trage concluzia falsă că poate acum să înceapă să-i transmită lui B.

În plus, există și problema inversă, a stației expuse, ilustrată în fig. 4-26(b). Aici B vrea să-i trimită lui C astfel încât observă canalul. Când aude o transmisiune trage concluzia falsă că nu poate să-i transmită lui C, chiar dacă de fapt A îi transmite lui D (care este absent din imagine).

Mai mult, majoritatea radiourilor sunt semi-duplex, ceea ce înseamnă că nu pot transmite și asculta zgomotele de coliziuni simultan pe aceeași frecvență. Ca urmare a acestor probleme, 802.11 nu utilizează CSMA/CD utilizat de Ethernet.

Pentru a face față acestor probleme, 802.11 suportă două tipuri de operații. Primul, denumit **DCF** (**Distributed Coordination Function**, rom: funcție de coordonare distribuită), nu folosește nici un fel de control central (fiind similar în această privință Ethernetului). Celălalt, denumit **PCF** (**Point Coordination Function**, rom: funcție de coordonare punctuală), folosește stația de bază pentru a coordona toată activitatea din celula sa. Toate implementările trebuie să poată susține DCF, dar PCF este opțional. Vom discuta acum aceste două modalități, alternativ.

Când este folosit DCF, 802.11 utilizează un protocol denumit **CSMA/CA** (**CSMA with Collision Avoidance**, rom: CSMA cu evitarea coliziunilor). În acest protocol se folosește atât observarea canalelor fizice, cât și observarea canalelor virtuale. CSMA/CA suportă două tipuri de operații. În prima metodă, atunci când o stație vrea să transmită, observă canalul. Dacă este liber, începe să transmită. Nu va mai asculta canalul în timpul transmisiunii, ci va trimite întregul cadru, care ar putea foarte bine să fie distrus la receptor datorită interferenței. Dacă în schimb canalul este ocupat, emițătorul amână transmisia până când mediul se eliberează și abia apoi începe să transmită. Dacă apare o coliziune, stațiile implicate așteaptă un timp aleatoriu, folosind algoritmul exponențial de regresie binară, și mai încearcă o dată.

Cealaltă metodă a CSMA/CA se bazează pe MACAW și folosește observarea canalelor virtuale, după cum este ilustrat în fig. 4-27. În acest exemplu, *A* vrea să transmită către *B*. *C* este o stație din sfera de recepție a lui *A* (posibil și a lui *B*, dar nu contează). *D* este o stație din sfera lui *B* dar în afara sferei lui *A*.

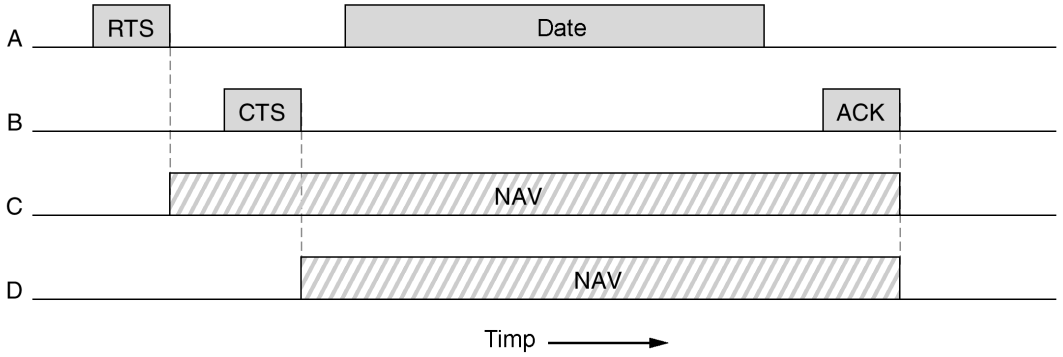


Fig. 4-27. Utilizarea canalului virtual utilizând CSMA/CA.

Protocolul începe când *A* decide că vrea să emită către *B*. Începe prin a trimite un cadru RTS către *B*, pentru a-i cere permisiunea să îi trimită un cadru. Când *B* primește această cerere, se poate decide să ofere permisiunea, caz în care trimite un cadru CTS înapoi. După ce primește CTS, *A* își trimite cadrul și inițiază un cronometru pentru ACK. După recepționarea corectă a cadrului de date,

B răspunde cu un cadru ACK, terminând schimbul. În cazul în care cronometrul pentru ACK a lui *A* expiră înainte ca ACK să revină la el, întregul protocol este luat de la capăt.

Să ne uităm acum la acest schimb din perspectiva lui *C* și a lui *D*. *C* este în sfera lui *A*, deci poate primi cadrul RTS. Dacă îl primește, își va da seama că o stație urmează să emită date în curând, astfel încât se va abține să transmită orice până când schimbul este complet. Din informația prezentă în cererea RTS poate estima durata tranzacției, incluzând ACK-ul final, astfel încât își simulează o ocupare virtuală, indicată în fig. 4-27 de **NAV** (**Network Allocation Vector**, rom: vector de alocare a rețelei). *D* nu aude RTS, dar aude CTS, astfel încât de asemenea își alocă un NAV. Observați că semnalele NAV nu sunt transmise; ele sunt doar modalități interne de a aminti stațiilor să tacă pentru o anumită perioadă.

Spre deosebire de rețelele cablate, rețelele fără fir sunt zgomotoase și instabile, într-o oarecare măsură și datorită cuptoarelor cu microunde, care folosesc și ele banda fără licență ISM. Ca urmare, probabilitatea unui cadru de a ajunge la destinație cu succes scade odată cu creșterea lungimii cadrului. Dacă probabilitatea unei erori la nivel de 1 bit este p, atunci probabilitatea unui cadru de n biți de a ajunge corect la final este $(1-p)^n$. De exemplu, dacă $p=10^{-4}$, probabilitatea de a recepționa corect un cadru Ethernet întreg (12.144 biți) este mai mică de 30%. Dacă $p=10^{-5}$, aproximativ un cadru din 9 va fi afectat. Chiar dacă $p=10^{-6}$, peste 1% dintre cadre vor fi afectate, ceea ce înseamnă aproape o duzină pe secundă, și chiar mai multe dacă se folosesc cadre mai scurte decât valoarea maximă. Pe scurt, dacă un cadru este prea lung, are puține șanse să fie transmis fără avarii, și va trebui, cel mai probabil, retransmis.

Pentru a rezolva problema canalelor zgomotoase, 802.11 permite cadrelor să fie fragmentate în bucăți mai mici, fiecare cu propria sumă de control. Fragmentele sunt numerotate individual și confirmate folosind un protocol pas-cu-pas (emițătorul nu poate transmite fragmentul k+1 decât după ce a primit confirmarea pentru fragmentul k). Odată ce canalul a fost obținut prin RTS și CTS, mai multe fragmente pot fi trimise într-un șir, ca în fig. 4-28, secvența purtând numele de **rafală de fragmente** (**fragment burst**).

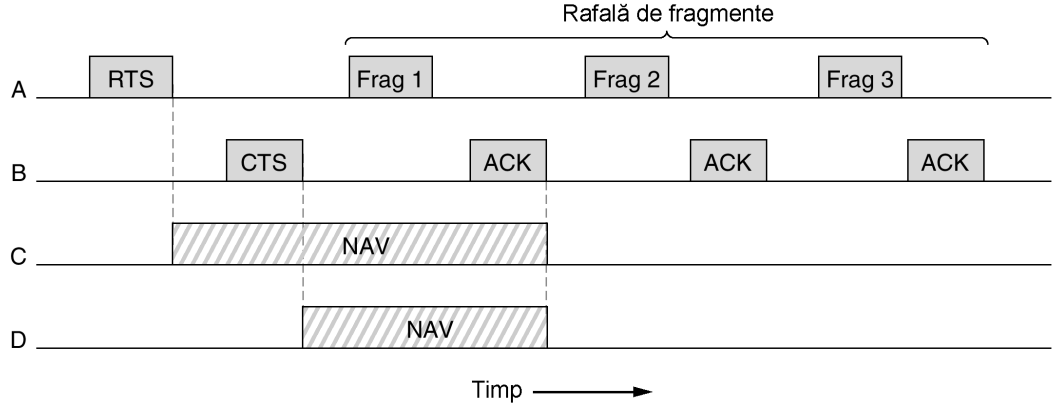


Fig. 4-28. O rafală de fragmente.

Fragmentarea crește productivitatea prin restricționarea retransmiterii doar la fragmentele eronate, eliminând necesitatea de a retransmite întregul cadru. Dimensiunea fragmentului nu este fixată de standard, ci este un parametru pentru fiecare celulă și poate fi ajustată de către stația de bază. Mecanismul NAV menține celelalte stații tăcute numai până la confirmarea următoare, dar un alt

mecanism (descris în cele ce urmează) este utilizat pentru a permite unei întregi rafale de fragmente să fie transmisă fără interferențe.

Toată discuția anterioară se aplică numai modului DCF 802.11. În acest mod, nu există control centralizat, iar stațiile sunt în competiție pentru timpul de transmisie, exact ca la Ethernet. Celălalt mod permis este PCF, în care stația de bază interoghează celelalte stații întrebându-le dacă au cadre de transmis. Din moment ce, în mod PCF, ordinea transmisiilor este complet controlată de către stația de bază, nu apar niciodată coliziuni. Standardul prescrie mecanismul pentru interogare, dar nu frecvența interogărilor, ordinea interogărilor și nici măcar dacă stațiile trebuie să primească toate drepturi egale.

Mecanismul principal este acela prin care stația de bază emite periodic (de 10 până la 100 de ori pe secundă) un **cadru baliză** (**beacon frame**). Cadrul baliză conține parametrii de sistem, cum ar fi intervalul de salt și timpii de viață (pentru FHSS), sincronizări de ceas, etc. De asemenea, acest cadru invită stațiile noi să se înregistreze pentru serviciul de interogare. Din momentul în care o stație s-a înregistrat pentru serviciul de interogare la o anumită viteză, aceasta va beneficia garantat de o fracțiune din lățimea de bandă, în acest fel fiind posibilă oferirea de garanții de tip calitatea-serviciului.

Viața bateriei este întotdeauna o problemă în cazul dispozitivelor mobile fără fir, deci 802.11 acordă atenție problemei gestionării consumului. În particular, stația de bază poate instrui un dispozitiv mobil să intre în așteptare până când este trezit în mod explicit de către stația de bază sau de către utilizator. Totuși, punerea unei stații în așteptare presupune ca stația de bază să aibă responsabilitatea stocării în zone tampon a cadrelor direcționate către stația respectivă atâta timp cât aceasta este în așteptare. Aceste cadre vor putea fi colectate ulterior.

Modurile PCF și DCF pot coexista în cadrul aceleiași celule. La prima vedere, pare imposibilă prezența unei scheme de control centralizate și a unei scheme distribuite în același timp, dar 802.11 oferă o modalitate de a atinge acest scop. Modul de funcționare este atins prin definirea atentă a intervalului de timp dintre transmisia de cadre. După ce un cadru a fost transmis, un anumit interval de timp este impus stației înainte de a putea transmite următorul cadru. Sunt definite patru intervale diferite, fiecare pentru scopuri precise. Cele patru intervale sunt prezentate în fig. 4-29.

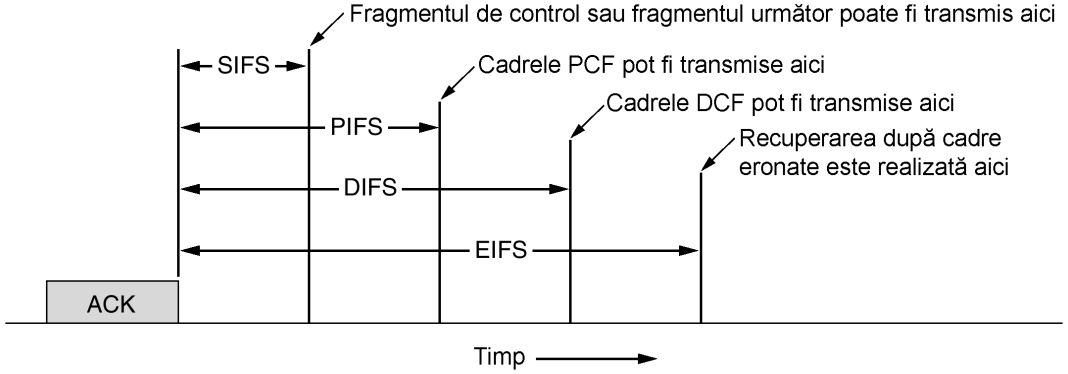


Fig. 4-29. Spațierea între cadre în 802.11.

Cel mai scurt interval este **SIFS** (**Short InterFrame Spacing**, rom: spațiere redusă între cadre). Este folosit pentru a permite părților implicate într-un dialog singular să transmită primele. Această soluție oferă posibilitatea receptorului să trimită un CTS ca răspuns la un RTS, acordă permisiunea ca un receptor să trimită o confirmare pentru un fragment sau pentru un cadru integral și acordă

permisiunea ca emițătorul unui fragment să trimită în rafală următorul fragment fără a fi nevoie să aștepte din nou un RTS.

Întotdeauna există exact o singură stație care are dreptul să răspundă după un interval SIFS. Dacă aceasta nu reușește să profite de această oportunitate și se scurge un interval de timp **PIFS (PCF InterFram Interval**, rom: interval între cadrele PCF), stația de bază poate transmite un cadru baliză sau un cadru de interogare. Acest lucru permite unei stații care transmite un cadru de date sau o secvență de fragmente să termine cadrul fără să se interpună nimeni, dar în același timp dă stației de bază o șansă să acapareze canalul atunci când emițătorul anterior a terminat, fără să trebuiască să intre în competiție cu utilizatorii nerăbdători.

Dacă stația de bază nu are nimic de transmis și se scurge un interval de timp **DIFS (DCF InterFrame Interval**, rom: interval între cadrele DCF), orice stație poate încerca obținerea unui canal pentru a transmite un nou cadru. Regulile obișnuite de competiție se aplică și o reluare după un timp binar exponențial poate fi necesară în cazul apariției unei coliziuni.

Ultimul interval de timp, **EIFS (Extended InterFrame Spacing**, rom: spațiere extinsă între cadre), este utilizat pentru a raporta eventuale erori de către o stație care tocmai a primit un cadru eronat sau necunoscut. Ideea este ca acest eveniment să aibă cea mai mică prioritate, pentru că, din moment ce receptorul s-ar putea să nu știe ce se întâmplă, acesta ar trebui să aștepte un interval substanțial de timp pentru a evita interferența cu un dialog în desfășurare între două stații.

4.4.4 Formatul cadrului 802.11

Standardul 802.11 definește trei clase diferite de cadre: de date, de control și de gestionare. Fiecare dintre acestea are un antet cu o varietate de câmpuri folosite în cadrul subnivelului MAC. În plus, există unele antete utilizate de către nivelul fizic, dar acestea sunt de obicei legate de modalitățile de modulație folosite, deci nu le vom discuta aici.

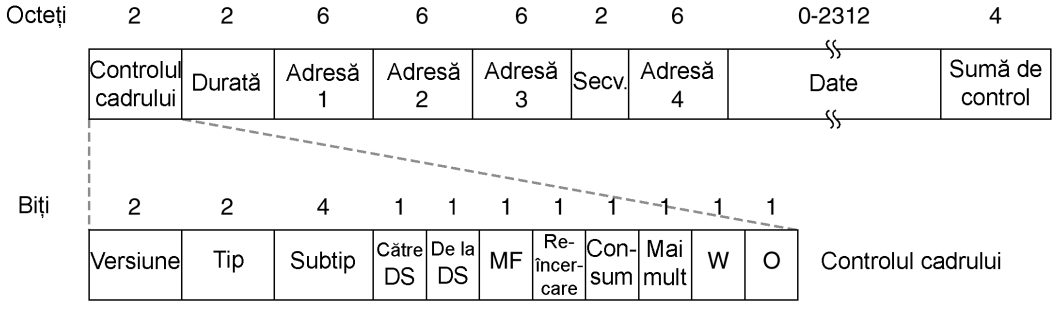


Fig. 4-30. Cadrul de date 802.11.

Formatul cadrului de date este prezentat în fig. 4-30. Primul câmp este acela de *control al cadrului*. Acesta are la rândul lui 11 subcâmpuri. Primul dintre acestea este *versiunea de protocol*, care permite celor două versiuni ale protocolului să opereze în același timp în aceiași celulă. Apoi urmează câmpurile pentru *tip* (de date, de control sau de gestiune) și *subtip* (de ex. RTS sau CTS). Biții *către DS* și *de la DS* indică direcția de transport a cadrului – către sistemul de distribuție sau de la sistemul de distribuție între celule (de ex. Ethernet). Bitul *MF* semnalizează că vor urma mai multe fragmente. Bitul *reîncercare* marchează o retransmisie a unui cadru trimis anterior. Bitul *gestiunea consumului* este folosit de către stația de bază pentru a pune receptorul în stare de așteptare sau pentru a-l scoate din

starea de așteptare. Bitul *mai mult* indică faptul că emițătorul mai are cadre adiționale pentru receptor. Bitul *W* specifică criptarea cadrului folosind algoritmul **WEP** (**Wired Equivalent Privacy**, rom. confidențialitate echivalentă cablată). În sfârșit, bitul *O* indică receptorului că o secvență de cadre cu acest bit setat trebuie prelucrată strict în ordinea în care cadrele au fost recepționate.

Cel de-al doilea câmp al cadrului de date, câmpul *durată*, indică intervalul de timp în care cadrul și confirmarea vor ocupa canalul. Acest câmp este de asemenea prezent în cadrele de control și reprezintă modalitatea prin care alte stații gestionează mecanismul NAV. Antetul cadrului conține patru adrese, toate în formatul standard IEEE 802. În mod evident, sursa și destinația sunt necesare, dar pentru ce sunt necesare celelalte două? Să ne aducem aminte: cadrele pot intra sau ieși dintr-o celulă prin stația de bază. Celelalte două adrese sunt folosite pentru stațiile de bază sursă și destinație în traficul între celule.

Câmpul *secvență* permite numerotarea fragmentelor. Din cei 16 biți disponibili, 12 identifică cadrul și 4 identifică fragmentul. Câmpul *date* conține încărcătura utilă, având până la 2312 octeți, fiind urmat de uzuala *sumă de control*.

Cadrele de gestiune au un format similar celor de date, cu excepția uneia dintre adresele stațiilor de bază, deoarece cadrele de gestiune sunt restricționate la o singură celulă. Cadrele de control sunt și mai scurte, având numai una sau două adrese, neavând câmp de *date* și nici câmp *secvență*. Aici informația cheie se află în câmpul *subtip*, de obicei RTS, CTS, sau ACK.

4.4.5 Servicii

Standardul 802.11 afirmă că fiecare rețea locală fără fir trebuie să ofere nouă servicii. Aceste servicii sunt împărțite în două categorii: cinci servicii legate de distribuție și patru servicii pentru stații. Serviciile de distribuție sunt legate de gestiunea apartenenței la celulă și de interacțiunea cu stațiile din afara celulei. Serviciile de stație sunt legate de activitatea în cadrul unei singure celule.

Cele cinci servicii de distribuție sunt oferite de către stațiile de bază și se ocupă de mobilitatea stațiilor pe măsură ce acestea intră și ies din celule, atașându-se și detașându-se de la stația de bază. Ele sunt următoarele:

1. **Asocierea**. Acest serviciu este folosit de către stațiile mobile pentru conectare la stația de bază. De obicei, el este utilizat exact după ce o stație se deplasează în acoperirea radio a stației de bază. La sosire, aceasta își anunță identitatea și capacitățile. Capacitățile includ: vitezele de date suportate, cererile de servicii PCF (de exemplu interogarea) și necesitățile de gestionare a consumului. Stația de bază poate accepta sau rejecta stația mobilă. Dacă stația mobilă este acceptată, atunci aceasta trebuie să se autentifice.
2. **Dezasocierea**. Atât stația, cât și stația de bază se pot dezasocia, rupând în acest fel relația. O stație trebuie să folosească acest serviciu înainte de a se închide sau de a pleca, iar stația de bază îl poate folosi și ea – de exemplu înainte de oprirea pentru întreținere.
3. **Reasocierea**. O stație își poate schimba stația de bază preferată utilizând acest serviciu. Facilitatea este utilă pentru stațiile mobile care se deplasează dintr-o celulă în alta. Dacă serviciul este folosit corect, nu se vor pierde date la trecere. (Dar 802.11, ca și Ethernetul, este numai un serviciu fără confirmare).
4. **Distribuția**. Acest serviciu determină modul în care sunt rutate cadrele trimise către stația de bază. Dacă destinația este locală stației de bază, cadrele pot fi trimise direct în aer. În caz contrar, ele vor trebui înaintate prin rețeaua cablată.

5. **Integrarea**. Dacă un cadru trebuie să circule printr-o rețea care nu este 802.11 și utilizează o schemă de adresare diferită și un format de cadre diferit, acest serviciu efectuează translatarea de la formatul 802.11 la formatul rețelei destinație.

Cele patru servicii rămase sunt servicii în interiorul celulei (adică sunt legate de acțiuni în interiorul unei singure celule). Ele sunt folosite după ce a avut loc asocierea și sunt următoarele:

1. **Autentificarea.** Deoarece comunicațiile fără fir pot fi recepționate sau emise cu ușurință de către stații neautorizate, o stație trebuie să se autentifice înainte de a i se permite să trimită date. După ce o stație mobilă a fost asociată de către o stație de bază (adică acceptată în cadrul celulei), stația de bază îi va trimite un cadru special de provocare pentru a verifica dacă stația mobilă cunoaște cheia secretă (parola) care i-a fost alocată. Stația va dovedi cunoașterea cheii prin criptarea cadrului provocare cu ea, urmată de trimiterea acestuia înapoi la stația de bază. Dacă rezultatul este corect, stația mobilă este pe deplin înscrisă în celulă. În standardul inițial, stația de bază nu trebuia să-și demonstreze identitatea către stațiile mobile, dar lucrul pentru eliminarea acestui defect din standard este în desfășurare.
2. **Deautentificarea.** Când o stație anterior autentificată dorește să părăsească rețeaua, aceasta este deautentificată. După deautentificare, stația nu va mai putea folosi rețeaua.
3. **Confidențialitatea.** Pentru ca informațiile transmise printr-o rețea fără fir să rămână confidențiale, acestea trebuie criptate. Acest serviciu gestionează criptarea și decriptarea. Algoritmul de criptare specificat este RC4, inventat de către Ronald Rivest de la M.I.T.
4. **Livrarea datelor.** În fine, transmisia datelor este subiectul principal, deci în mod evident 802.11 oferă o modalitate de a transmite și recepționa date. Din moment ce 802.11 este modelat după Ethernet și transmisia prin Ethernet nu este 100% sigură, transmisiile prin 802.11 nu sunt garantate nici ele să fie sigure. Nivelurile superioare trebuie detecteze și să corecteze erorile.

O celulă 802.11 are unii parametrii care pot fi inspectați și, în unele cazuri, ajustați. Aceștia sunt legați de criptare, intervale de expirare, viteze de transfer pentru date, frecvența balizelor și așa mai departe.

Rețelele locale fără fir bazate pe 802.11 încep să fie folosite peste tot în lume în clădiri de birouri, hoteluri, restaurante și campusuri. Se preconizează o creștere rapidă. Pentru unele date legate de desfășurarea extinsă a 802.11 la CMU, vezi (Hills, 2001).

4.5 REȚELE FĂRĂ FIR DE BANDĂ LARGĂ

Am stat în clădiri prea mult. Haideți să ieșim afară și să vedem dacă există rețele interesante acolo. Se pare că se întâmplă destul de multe pe acolo și o parte dintre acestea au de-a face cu așa numită ultimă-porțiune (ultima sută de metri). Odată cu deregularizarea serviciilor telefonice în multe țări, competitori ai companiilor telefonice fortificate au din ce în ce mai des permisiunea să ofere servicii de voce și de Internet la mare viteză. Există în mod sigur o mare cerere în domeniu. Problema este că desfășurarea de fibră, de cablu coaxial sau chiar de perechi torsadate de categoria 5 este prohibitiv de scumpă. Ce poate face atunci un competitor?

Răspunsul este simplu: rețele fără fir de bandă largă. Ridicarea unei antene mari pe un deal imediat în afara orașului și instalarea de antene direcționate către ea pe acoperișurile clienților este mult mai simplu și mai ieftin decât săparea de șanțuri și tragerea de cabluri prin acestea. Prin urmare, companiile de telecomunicații concurente au un interes important în a oferi servicii de comunicație fără fir la viteze de mulți megabiți pe secundă pentru voce, Internet, filme la cerere etc. După cum am văzut în fig. 2-30, LMDS a fost inventat pentru acest scop. Totuși, până de curând, fiecare companie și-a dezvoltat propriul sistem. Această lipsă de standarde a însemnat că software-ul și hardware-ul nu au putut fi produse pe scară largă, ceea ce a menținut prețurile la un nivel mare și acceptabilitatea la un nivel scăzut.

Mulți oameni din industrie au realizat faptul că existența unui standard pentru rețele fără fir de bandă largă era elementul cheie care lipsea, așa că organizației IEEE i s-a cerut să alcătuiască un comitet format din oameni din companii cheie și din mediul academic pentru a schița standardul. Următorul număr disponibil în schema de numerotare 802 era **802.16**, deci standardul a primit acest număr. Lucrul a început în iulie 1999, iar standardul final a fost acceptat în aprilie 2002. Oficial, standardul poartă numele "Interfață aeriană pentru acces fix la sisteme cu lățime mare de bandă fără fir". Totuși, unii preferă să îl numească **wireless MAN** (**Wireless Metropolitan Area Network**, rom. rețea de întindere metropolitană fără fir) sau **wireless local loop** (rom. buclă locală fără fir). Noi vom privi toți acești termeni ca fiind interschimbabili.

La fel ca și alte standarde din seria 802, 802.16 a fost puternic influențat de modelul OSI, incluzând (sub)nivelurile, terminologia, primitivele de servicii și altele. Din păcate, ca și standardul OSI, este destul de complicat. În următoarele secțiuni vom oferi o descriere sumară a lui 802.16, dar tratarea de aici este departe de a fi completă și lasă neacoperite multe detalii. Pentru informații adiționale despre servicii fără fir de bandă largă în general, vezi (Bolcskei et. al, 2001; și Webb, 2001). Pentru informații specifice despre 802.16, vezi (Eklund et al., 2002).

4.5.1 Comparație între 802.11 și 802.16

În acest moment probabil că vă întrebați: De ce să inventăm un nou standard? De ce nu folosim pur și simplu 802.11? Există câteva motive foarte bune pentru a nu folosi 802.11, în primul rând acela că 802.11 și 802.16 rezolvă probleme diferite. Înainte de a intra în terminologia lui 802.16, probabil că merită să spunem câteva cuvinte pentru a explica de ce este nevoie de un nou standard.

Mediile în care operează 802.11 și 802.16 sunt similare în multe privințe, în primul rând datorită faptului că sunt proiectate pentru a oferi comunicații fără fir de mare viteză. Dar de asemenea diferă din multe puncte de vedere majore. Pentru început, 802.16 oferă servicii pentru clădiri, iar clădirile nu sunt mobile. Ele nu migrează dintr-o celulă în alta. Mare parte din 802.11 tratează mobilitatea și o mare parte dintre aceste aspecte nu sunt relevante aici. Apoi, clădirile dispun de mai multe calculatoare, o complicație care nu apare atunci când stația finală este un singur calculator portabil. Pentru că proprietarii clădirilor sunt de obicei dispuși să cheltuiască mult mai mulți bani pe echipamente de comunicații decât proprietarii de calculatoare portabile, vor fi disponibile stații radio mai performante. Această diferență înseamnă și că 802.16 poate folosi comunicații cu duplex integral, lucru evitat de 802.11 pentru a menține prețurile la un nivel scăzut.

Pentru că 802.16 rulează peste o parte dintr-un oraș, distanțele implicate pot fi de mai mulți kilometri, ceea ce înseamnă că puterea absorbită la stația de bază poate varia mult de la o stație la alta. Această variație afectează raportul semnal-zgomot, care, la rândul lui, dictează mai multe scheme de

modulare. În același timp, comunicare deschisă peste un oraș înseamnă că securitatea și confidențialitatea sunt esențiale și obligatorii.

Mai mult, este foarte probabil ca fiecare celulă să aibă mult mai mulți utilizatori decât o celulă tipică 802.11 și este de așteptat ca acești utilizatori să consume o lățime de bandă mult mai mare decât utilizatorii tipici ai 802.11. Până la urmă, nu se întâmplă des ca o companie să invite 50 de angajați, cu calculatoare portabile, să se întâlnească într-o singură cameră pentru a vedea dacă se poate satura rețeaua 802.11 prin vizionarea în paralel pe a 50 de filme diferite. Din acest motiv, este nevoie de un spectru mai larg decât oferă banda ISM, forțând 802.16 să opereze într-o gamă de frecvențe mult mai înalte, de 10-66 GHz, singurul loc în care gama de frecvențe este încă disponibilă.

Dar aceste unde milimetrice au proprietăți fizice diferite decât undele mai lungi din ISM, ceea ce duce la necesitatea unui nivel fizic complet diferit. O proprietate a undelor milimetrice este aceea ca sunt puternic absorbite de apă (în mod special de ploaie, dar până la un anumit nivel și de către ninsoare, grindină și, cu puțin ghinion, de ceață deasă). În consecință, tratarea erorilor este mult mai importantă decât în mediile interioare. Undele milimetrice pot fi focalizate în fascicule direcționate (802.11 este omnidirecțional), deci soluțiile alese de 802.11 pentru propagarea multicăi sunt contestabile aici.

O altă problemă o constituie calitatea serviciului. Deși 802.11 oferă suport pentru trafic de timp real (folosind modul PFC), acesta nu a fost proiectat pentru telefonie și nici pentru utilizare multimedia intensivă. Dimpotrivă, ne așteptăm ca 802.16 să ofere suport complet pentru aceste aplicații, deoarece este gândit să fie folosit atât pentru mediul rezidențial, cât și pentru mediul de afaceri.

Pe scurt, 802.11 a fost proiectat pentru a fi un Ethernet mobil, pe când 802.16 a fost proiectat pentru a fi o televiziune prin cablu, fără fir, dar staționară. Aceste diferențe sunt atât de mari, încât standardele rezultate sunt foarte diferite: ele încearcă să optimizeze lucruri diferite.

O comparație foarte sumară cu sistemul de telefonie celulară este de asemenea interesantă. La telefoanele mobile avem de-a face cu stații mobile de bandă îngustă, orientate pe voce, cu putere scăzută, care comunică folosind unde de lungime medie. Nimeni nu urmărește (încă) filme de 2 ore, la rezoluție înaltă, pe telefoanele GSM. Chiar și UMTS are speranțe scăzute în a schimba această situație. Pe scurt, lumea rețelelor metropolitane fără fir este mult mai solicitantă decât lumea simplă a telefoanelor mobile, deci este necesar un sistem complet diferit. Întrebarea interesantă este dacă 802.16 poate fi folosit pentru dispozitive mobile în viitor. Nu a fost optimizat pentru această utilizare, dar posibilități există. Pentru moment, standardul se concentrează pe rețele fixe fără fir.

4.5.2 Stiva de protocoale 802.16

Stiva de protocoale 802.16 este prezentată în fig. 4-31. Structura generală este similară cu cea a celorlalte rețele 802, dar are mai multe subniveluri. Subnivelurile inferioare se ocupă de transmisie. Radioul tradițional de bandă îngustă folosește scheme de modulare convenționale. Deasupra nivelului de transmisie fizic este un subnivel de convergență pentru a ascunde diferitele tehnologii de nivelul legătură de date. De fapt și 802.11 are ceva asemănător, dar comitetul a decis să nu-l formalizeze cu un nume asemănător OSI.

Cu toate că nu le-am prezentat în figură, două noi protocoale de nivel fizic sunt în lucru deja. Standardul 802.16a va oferi suport pentru OFDM în gama de frecvențe 2-11 GHz. Standardul 802.16b va lucra în banda ISM de 5 GHz. Ambele constituie încercări de apropiere de 802.11.

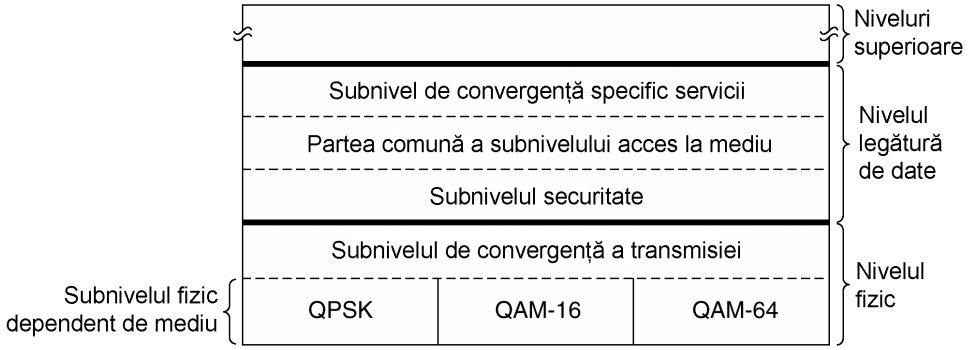


Fig. 4-31. Stiva de protocoale 802.16.

Nivelul legătură de date este constituit din trei subniveluri. Cel mai de jos se ocupă de confidențialitate și securitate, aspecte mult mai importante pentru rețelele publice de exterior decât în cadrul rețelelor private de interior. Acest subnivel gestionează cheile, criptarea și decriptarea.

Următorul este subnivelul comun de acces la mediu. Aici sunt localizate protocoalele principale, cum ar fi cel de gestiune a canalelor. Modelul presupune ca stația de bază să controleze sistemul. Aceasta poate planifica canalul către abonat (cu baza la abonat) în mod foarte eficient și joacă de asemenea un rol major în gestionarea canalelor de la abonat (adică abonat către bază). O facilitate neobișnuită a subnivelului de acces la mediu este aceea că, spre deosebire de toate celelalte rețele 802, este complet orientat pe conexiune, pentru a putea garanta calitatea serviciilor pentru comunicații multimedia și pentru telefonie.

Subnivelul de convergență joacă rolul subnivelului de legătură logică din celelalte protocoale 802. Scopul său este interfațarea cu nivelul rețea. O complicație în acest caz este aceea că 802.16 a fost proiectat pentru a integra fără diferențiere atât protocoale cu datagrame (de ex. PPP, IP și Ethernet), cât și ATM. Problema este că protocoalele cu datagrame sunt fără conexiune, pe când ATM-ul este orientat conexiune.

Aceasta înseamnă că fiecare legătură ATM trebuie să fie suprapusă peste o legătură 802.16, în cel mai direct mod posibil. Totuși, peste care legătură 802.16 ar trebui suprapus un pachet IP recepționat? Această problemă este rezolvată în cadrul acestui subnivel.

4.5.3 Nivelul fizic 802.16

Așa cum am menționat și mai înainte, transmisia fără fir de bandă largă are nevoie de un spectru larg și singura posibilitate de a obține acest spectru este în zona 10-66 GHz. Astfel de unde milimetrice prezintă o proprietate interesantă, care nu există la undele mai lungi: ele se propagă în linii drepte, diferit față de sunet, însă foarte asemănătoare cu lumina. O consecință directă este faptul că o stație de emisie poate avea multiple antene, fiecare îndreptată spre o zonă diferită a ariei de acoperire, așa cum este prezentat în fig. 4-32. Fiecare sector are proprii utilizatori și este independent într-o mare măsură de zonele adiacente, fapt care nu este prezent și în cazul sistemelor cu celule radio, unde transmisia este în toate direcțiile.

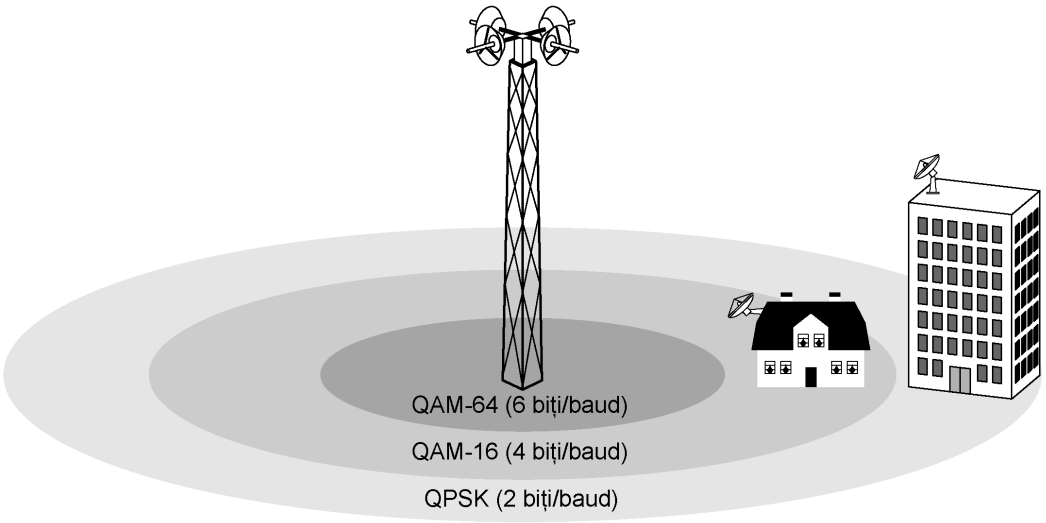


Fig. 4-32. Transmisia la 802.16.

Cum puterea semnalului este milimetrică, acoperirea scade proporțional cu distanța de la stația de emisie, raportul semnal/zgomot scade de asemenea, proporțional cu aceeași distanță. Din acest motiv, 802.16 are trei scheme de modulare diferite, în funcție de cât de departe este stația abonat de stația de emisie. Pentru abonații foarte apropiați de stația de emisie, este folosit QAM-64, cu 6 biți/baud. Pentru o distanță medie, este folosit QAM-16, cu 4 biți/baud. QPSK este folosit pentru abonații aflați la o distanța considerabilă și suportă 2 biți/baud. Spre exemplu, pentru un spectru tipic de 25 MHz, QAM-64 oferă 150 Mbps, QAM-16 doar 100 Mbps, în timp ce QPSK atinge până la 50 Mbps. Cu alte cuvinte, cu cât abonatul se află la o distanță mai mare de stația de emisie, cu atât viteza de transmisie scade (este asemănător cu ceea ce am văzut la ADSL, fig. 2-27). Diagramele sub formă de constelație pentru cele trei tipuri de modulație au fost prezentate în fig. 2-25.

Având ca scop crearea unui sistem de bandă largă în prezența limitărilor fizice descrise mai sus, designerii lui 802.16 au lucrat din greu la folosirea eficientă a spectrului disponibil. Unul din lucrurile pe care nu le-au apreciat a fost modul în care lucrează sistemele GSM și DAMPS. Amândouă utilizează, pentru traficul de recepție și pentru cel de emisie, benzi de frecvență diferite, egale ca dimensiune (simetrice). Dacă pentru voce traficul este simetric în cea mai mare parte, pentru accesul la Internet traficul este mult mai puternic la recepție decât la transmisie. Din aceste considerente, 802.16 oferă o modalitate mult mai flexibilă de a aloca banda de transfer. Sunt folosite două scheme, **FDD** (**Frequency Division Duplexing**, rom: transmisie duplex prin **d**ivizarea în frecvență) și **TDD** (**Time Division Duplexing – t**ransmisie duplex prin divizarea în timp). A doua schemă este prezentată în fig. 4-33. În acest caz, stația de emisie trimite periodic cadre. Fiecare cadru conține unități de timp. Primele sunt folosite la traficul de recepționare. Urmează o perioadă de timp de gardă, folosită de stații pentru a schimba direcția de transmisie. În cele din urmă avem intervalul în care se transmit datele locale. Numărul de cuante de timp alocat fiecărui tip de transmisie poate fi modificat dinamic astfel încât banda de transfer să fie folosită cât mai eficient.

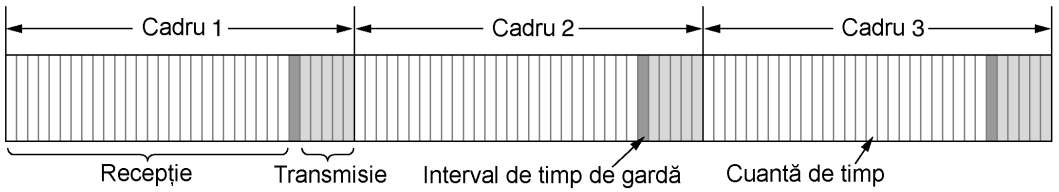


Fig. 4-33. Cadre și intervale de timp în TDD.

Traficul de recepție este suprapus peste cuantele de timp de către stația de emisie. Stația de emisie are controlul deplin pentru acest trafic. Traficul de transmisie este mult mai complex și depinde de calitatea cerută de serviciul accesat. Vom discuta despre alocarea intervalelor de timp când vom prezenta subnivelul MAC, puțin mai jos.

O altă caracteristică interesantă a nivelului fizic este posibilitatea de a împacheta mai multe cadre MAC într-o singură transmisie fizică. Aceasta crește eficiența, prin reducerea numărului total de preambuluri și antete folosite.

De notat este și folosirea codurilor Hamming pentru corecția în avans la nivelul fizic. Aproape toate celelalte tipuri de rețele se bazează pur și simplu pe trimiterea unor sume de control pentru detectarea erorilor, iar apoi cer retransmisia dacă unul din cadre a fost recepționat eronat. În cazul transmisiilor de bandă largă sunt estimate atât de multe erori, încât corectarea erorilor se face chiar la nivelul fizic, pe lângă sumele de control de la nivelurile superioare. În ansamblu, efectul pe care îl au toate acestea este de a face canalul să pară mai bun decât este în realitate (în același mod în care CD-ROM-urile par a fi foarte fiabile, și toate acestea doar pentru că mai mult de jumătate din numărul total de biți sunt folosiți la corectarea de erori la nivelul fizic).

4.5.4 Protocolul subnivelului MAC la 802.16

Nivelul legătură de date este împărțit în trei subniveluri, așa cum am văzut în fig. 4-31. Cum nu vom studia criptografia înainte de cap. 8, este mai dificil de explicat acum modul de funcționare a subnivelului de securitate. Este suficient însă să afirmăm că tehnicile de criptare sunt folosite pentru a păstra toate datele transmise secrete. Antetele nu sunt criptate, ci doar informația utilă. Aceasta înseamnă că cineva poate să urmărească cine cu cine comunică, dar nu poate înțelege conținutul mesajelor.

Dacă aveți deja câteva cunoștințe despre criptografie, urmează un scurt paragraf ce explică subnivelul de securitate. Dacă nu cunoașteți nimic despre criptografie, este puțin probabil să găsiți următorul paragraf foarte folositor (însă îl puteți lua în considerare după ce ați parcurs cap. 8).

În momentul în care un abonat se conectează la o stație de emisie, el execută o autentificare mutuală folosind RSA cu cheie publică și certificatele X.509. Datele utile sunt criptate folosind sistemul de chei simetrice, ori DES triplu, ori folosind înlănțuirea de blocuri cu cifru. Algoritmul AES (Rijndael) e posibil să fie adăugat în curând. Verificările de integritate sunt făcute folosind SHA-1. Nu e așa că nu a fost chiar atât de greu?

Haideți să aruncăm o privire peste partea principală a subnivelului MAC. Cadrele MAC ocupă un număr întreg de perioade de timp ale nivelului fizic. Fiecare cadru este format din subcadre, dintre care primele două reprezintă mapările pentru traficul de recepție și de transmisie. Aceste mapări relevă ce corespunde fiecărui interval de timp și ce cuante de timp sunt libere. Maparea pentru recepție conține și un număr de parametri de sistem, pentru a informa noile stații cu care intră în contact.

Canalul pentru recepție este destul de simplu, stația de emisie decide ce conține fiecare subcadru. Canalul de transmisie este mai complicat deoarece mai mulți abonați intră în competiție pentru acces. Alocarea acestui canal este legată de problemele de calitate a serviciului. Sunt definite patru clase de servicii:

1. Serviciu cu viteză constantă de transmisie
2. Serviciu pentru aplicații de timp real
3. Serviciu pentru aplicații care nu necesită timp real
4. Serviciu de tip cea mai bună încercare (best-effort)

Toate serviciile lui 802.16 sunt orientate pe conexiune, iar fiecare dintre conexiuni este stabilită la configurare la unul din tipurile de mai sus. Arhitectura este mult diferită de 802.11 și Ethernet, care nu prezintă conexiuni la subnivelul MAC.

Serviciul cu viteză constantă de transmisie este folosit pentru transmisie de voce, necompresată, la fel ca pe canalele T1. Acest serviciu are nevoie să trimită o valoare predeterminată de date, la un interval de timp stabilit apriori. Fiecărei conexiuni de acest tip îi este dedicat un număr de intervale de timp. Odată banda de transfer alocată, intervalele de timp sunt puse la dispoziție automat, fără a mai fi nevoie să fie cerute spre alocare.

Serviciul de aplicații de timp real este destinat aplicațiilor media compresate sau altor aplicații software de timp real, în care nevoia de bandă de transfer poate varia. Intră în atribuțiile stației de emisie să îl întrebe pe abonat, la un interval fixat de timp, de câtă bandă de transfer are nevoie.

Serviciul pentru aplicații ce nu necesită timp real este utilizat pentru transmisii mari de date, cum ar fi transferurile de fișiere mari ca dimensiuni. În acest caz, stația de emisie interoghează abonatul des, însă nu la intervale fixate de timp. Un abonat care transmite constant date poate cere, prin intermediul cadrelor sale, o interogare din partea stației de emisie, pentru a putea trimite date suplimentare.

Dacă o stație nu răspunde la o interogare repetată de k ori într-un anumit interval de timp, stația de emisie o va trece pe aceasta într-un grup căruia îi transmite unitar și îi anulează dreptul la o interogare individuală. Când un astfel de grup este interogat, oricare dintre stații poate răspunde, toate intrând în competiție pentru serviciul cerut. În acest fel, stațiile cu un trafic mic nu consumă inutil interogările stației de emisie.

În cele din urmă, serviciul fără garanție este valabil pentru toate celelalte cazuri. În acest caz nu există interogări, iar abonații trebuie să intre în competiție directă pentru servicii. Cererile de bandă de transfer sunt făcute în acele cuante de timp marcate ca fiind libere în cadrele primite în zona de transmisie pentru conectări. Dacă o cerere a fost satisfăcută, aceasta va fi notată în cadrul următor, în zona de mapare a recepționărilor. Abonatul va trebui să reîncerce conectarea în caz că cererea nu a fost satisfăcută. Pentru a minimaliza coliziunile este folosit algoritmul de regresie exponențială de la Ethernet.

Standardul definește două forme de alocare a benzii de transfer: pentru stație și pentru conexiune. În primul caz, spre exemplu, abonatul face cerere de bandă de transfer în numele tuturor utilizatorilor dintr-o clădire. Când banda de transfer îi este acordată, abonatul va acorda o parte din aceasta fiecărui utilizator, în funcție de cererile primite de el apriori. În forma a doua de alocare a benzii de transfer (pentru fiecare conexiune), stația de emisie se ocupă de fiecare conexiune în mod direct.

4.5.5 Structura cadrului 802.16

Toate cadrele MAC încep cu un antet generic. Antetul este urmat, opțional, de datele utile și tot opțional de o sumă de control (CRC), așa cum este ilustrat în fig. 4-34. Prezența informației utile nu este necesară în cadrele de control, cum sunt de exemplu cadrele pentru alocarea canalelor. Suma de control este și ea opțională datorită existenței corecției de erori la nivelul fizic și datorită faptului că nu se încearcă niciodată retransmiterea cadrelor de timp real. Dacă nu se folosește retransmisia, de ce să ne complicăm cu o sumă de control?

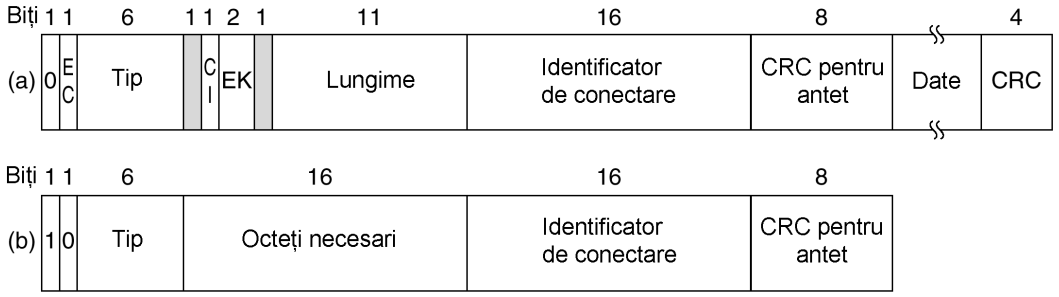

Biți 1 1 6 1 1 2 1 11 16 8 4
(a) 0 EC Tip CI EK Lungime Identificator de conectare CRC pentru antet Date CRC
Biți 1 1 6 16 16 8
(b) 1 0 Tip Octeți necesari Identificator de conectare CRC pentru antet

Fig. 4-34. (a)Cadru generic. (b)Cadru de cererea a benzii de transmisie.

Urmează o scurtă descriere asupra câmpurilor din fig. 4-34(a). Bitul *EC* arată dacă informația utilă este sau nu criptată. Câmpul *Tip* identifică tipul cadrului și, în esență, informează dacă este prezentă împachetarea sau fragmentarea. Câmpul *CI* indică absența sau prezența sumei de control. În câmpul *EK* se menționează care dintre cheile de criptare este folosită. Câmpul *Lungime* oferă informații despre lungimea completă a cadrului, incluzând și antetul. Identificatorul de conectare arată cărei conexiuni aparține acest cadru. În cele din urmă, câmpul CRC pentru antet reprezintă o sumă de control aplicată doar antetului, folosind polinomul x^8+x^2+x+1.

Al doilea tip de antet, cel din fig. 4-34(b), este folosit pentru cererile de bandă de transfer. El începe cu primul bit de valoare 1, în loc de 0, și este similar cu cadrul generic, cu excepția faptului că al doilea și al treilea octet formează un număr de 16 biți, care precizează mărimea benzii de transfer necesară. Cadrele de cerere de bandă de transfer nu poartă nici o informație utilă și nici un câmp CRC.

Ar mai fi multe de spus despre 802.16, însă nu mai este loc pentru aceasta. Prin urmare, pentru mai multe informații, vă rugăm să consultați standardul.

4.6 BLUETOOTH

În 1994, compania L. M. Ericsson a început să fie interesată în a conecta telefoanele mobile pe care le producea cu alte dispozitive (de exemplu, PDA) fără a folosi cabluri. Împreună cu alte patru companii (IBM, Intel, Nokia și Toshiba) a creat un SIG (Special Interest Group, rom: grup special de interes sau consorțiu) pentru a dezvolta un standard de comunicație fără fir pentru interconectarea dispozitivelor de calcul și comunicare și a accesoriilor folosind frecvențe radio pe distanțe scurte care beneficiază de avantajul de a fi o tehnologie ieftină și fără un consum mare de putere. Proiectul

a fost numit **Bluetooth** după numele regelui viking Harald Blaatand (Bluetooth), ce a "unificat" (cucerit) Danemarca și Norvegia, desigur tot fără cabluri.

Deși ideea inițială era doar aceea de a scăpa de cablurile dintre dispozitive, încetul cu încetul standardul s-a dezvoltat și a intrat în scurt timp în aria rețelelor locale fără fir. Pe de o parte, această schimbare are avantajul de a face standardul mai util, dar pe de altă parte se creează o competiție cu 802.11. Mai mult decât atât, cele două sisteme interferează electric între ele. Este bine de menționat ca Hewlett-Packard a introdus o rețea infraroșu pentru conectarea fără fire a perifericelor în urmă cu câțiva ani, dar nu a ajuns foarte departe cu aceasta.

Fără a fi descurajat de acest fapt, în iulie 1999, SIG Bluetooth a publicat specificația 1.0, având aproximativ 1500 de pagini. La scurt timp, grupul de standardizare IEEE a preluat standardul Bluetooth în aria sa de standarde pentru rețele personale fără fir și a început să îl ajusteze. Deși pare ciudat să se încerce să se standardizeze ceva ce a fost deja foarte bine detaliat în specificații și pentru care nu există nici o incompatibilitate care să aibă nevoie de armonizare, istoria arată că redactarea unui standard de către o entitate neutră, așa cum este IEEE, a dus de cele mai multe ori la promovarea tehnologiei respective. Pentru a fi mai exacți, trebuie menționat că specificația Bluetooth este elaborată pentru întregul sistem, de la nivelul fizic și până la nivelul aplicație. Comitetul 802.15 standardizează numai nivelul fizic și nivelul de legătură de date, restul stivei de protocoale nefiind în afara atenției sale.

Chiar dacă IEEE a aprobat în 2002 primul standard pentru rețele personale 802.15.1, SIG Bluetooth este încă foarte activ la îmbunătățirea standardului. În ciuda faptului că versiunile emise de SIG Bluetooth și IEEE nu sunt identice, se speră că în curând cele două vor converge spre un standard comun.

4.6.1 Arhitectura Bluetooth

Haideți să începem studiul nostru asupra sistemului Bluetooth cu o scurtă prezentare de ansamblu asupra a ceea ce conține și ce este proiectat să facă. Unitatea de bază a unui sistem Bluetooth este un **piconet** (**pico rețea**) format dintr-un nod stăpân (master) și până la 7 noduri sclav (slave), toate într-o regiune cu diametrul maxim de 10 metri. Mai multe piconet-uri pot exista în aceeași încăpere și chiar pot fi conectate printr-un nod de trecere, așa cum arată fig. 4-35. O colecție interconectată de piconet-uri este denumită **scatternet** (**rețea dispersată**).

În afara celor șapte noduri sclav active dintr-un piconet, în rețea pot exista până la 255 de noduri în modul parcat. Acestea sunt dispozitive pe care stăpânul le-a trecut în modul de consum redus, economisind astfel consumul de putere de la baterii. În modul parcat, un dispozitiv nu poate face altceva decât să răspundă la semnalele temporare ale stăpânului sau la cele de activare. De asemenea, există încă două stări intermediare: așteptare și „ascultare" (eng. sniff), însă ele nu vor fi discutate în această parte.

Motivul pentru care s-a ales arhitectura stapân/sclav este acela că designerii au intenționat să faciliteze o implementare completă a cipurilor Bluetooth sub 5$. Consecința acestei decizii este aceea că sclavii sunt destul de limitați, și, în esență, ei fac doar ceea ce le spune stăpânul să facă. Ca principiu de bază, un piconet este un sistem TDM centralizat, în care stăpânul controlează ceasul și determină care dispozitiv primește dreptul de a comunica și pentru ce perioadă de timp. Toate comunicațiile sunt între stăpân și sclav; nu pot exista comunicații directe sclav-sclav.

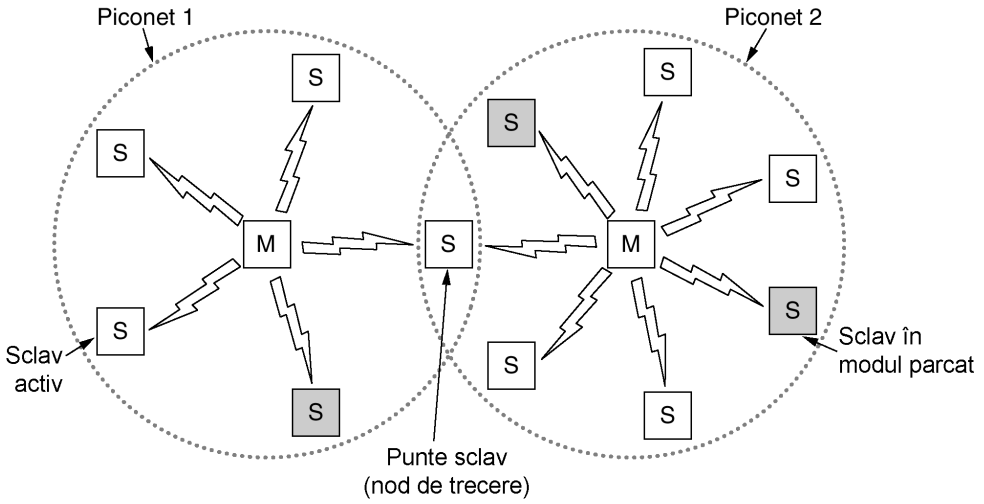


Fig. 4-35. Doua piconet-uri pot fi conectate să formeze o rețea dispersată (scatternet).

4.6.2 Aplicații Bluetooth

Majoritatea protocoalelor de rețea oferă doar canale între entitățile comunicante și lasă designerii de aplicații să decidă în ce mod vor să le folosească. Spre exemplu, 802.11 nu specifică dacă utilizatorii ar trebui să folosească notebook-urile personale pentru a-și citi emailul, pentru a naviga pe internet, sau orice altceva. Spre deosebire de acestea, specificația Bluetooth V1.1 definește 13 aplicații specifice care sunt suportate și oferă stive diferite de protocoale pentru fiecare dintre ele. Din păcate, această abordare este destul de complexă și de aceea în această carte vom încerca să simplificăm anumite aspecte. Cele 13 aplicații, denumite **profiluri**, sunt enumerate în fig. 4-36. Uitându-ne sumar la ele, putem să determinăm mult mai ușor care sunt intențiile SIG Bluetooth.

Denumire	Descriere
Acces generic	Proceduri pentru întreținerea legăturii
Descoperire de servicii	Protocol de descoperire a serviciilor oferite
Port serial	Înlocuitor pentru cablul de port serial
Intershimbare generică a obiectelor	Definește relația client-server pentru vehicularea de obiecte
Acces la rețeaua locală	Protocol între un calculator mobil și o rețea fixă
Rețea pe linie telefonică	Oferă posibilitatea ca un notebook să se apeleze folosind un telefon mobil
Fax	Permite unui fax mobil să comunice cu un telefon mobil
Telefonie fără fir	Conectează un set de căști de stația sa locală de emisie
Emitator-Receptor portabil (Intercom)	Radio-telefon portabil digital
Căști de telefon cu transmițător	Permite transmisii de voce hands-free (fără folosirea mâinilor)
Trimitere a obiectelor	Oferă o modalitate de schimbare a obiectelor simple
Transfer de fișiere	Oferă o facilitate mai generală de transfer de fișiere
Sincronizare	Oferă posibilitatea unui PDA să se sincronizeze cu un calculator

Fig. 4-36. Profiluri Bluetooth.

Profilul de acces generic nu este o aplicație în sine, ci este mai degrabă o bază pe care se pot construi aplicațiile reale. Principala sa atribuție este să ofere posibilitatea de a crea și a menține

conexiuni (canale) sigure între stăpân și sclav. De asemenea, profilul de descoperire a serviciilor este relativ generic. El este folosit de dispozitive pentru a descoperi ce servicii sunt oferite de către alte stații din rețea. Toate dispozitivele Bluetooth trebuie să implementeze aceste două profiluri. Celelalte rămân opționale.

Profilul de port serial este un protocol de transport care este folosit de majoritatea celorlalte profiluri. Acesta emulează o linie serială și este foarte util în special pentru aplicațiile mai vechi care necesit o astfel de facilitate.

Profilul de interschimbare generică a obiectelor definește o relație client-server pentru transmisia de date. Clienții inițiază operații, însă un sclav poate fi sau client sau server. Ca și profilul de port serial, este un punct de pornire pentru celelalte profiluri.

Următorul grup de trei profiluri este pentru lucrul în rețea. Profilul de acces la rețeaua locală permite unui dispozitiv Bluetooth să se conecteze la o rețea specificată. Acest profil este un competitor direct cu 802.11. Profilul de dial-up de rețea a fost motivația inițială a întregului proiect. El permite ca un notebook să se conecteze fără fire la un telefon mobil ce conține un modem intern. Profilul de fax este similar cu cel de dial-up de rețea, cu diferența că permite ca un fax neconectat la rețeaua de telefonie să trimită și să primească faxuri folosind telefoane mobile, fără existența unui fir între cele două.

Următoarele trei profiluri sunt pentru telefonie. Profilul de telefonie fără fir oferă o modalitate de a conecta un set de căști fără fir la stația de telefon. Momentan, majoritatea telefoanelor fără fir nu pot fi folosite și ca telefoane mobile, dar în viitor, este posibil ca telefoanele mobile și cele fără fir să devină unul și același lucru. Profilul Emițător-Receptor portabil (intercom) permite ca două telefoane să fie interconectate ca radiotelefoane portabile (walkie-talkie). În sfârșit, profilul pentru căști oferă comunicații de voce între căști cu transmițător și stația de bază a telefonului, spre exemplu, pentru comunicații fără intervenție manuală (hands-free) în timpul condusului mașinii.

Ultimele trei profiluri sunt folosite pentru schimburi de obiecte între două dispozitive fără fir. Acestea ar putea fi cărți de vizită, poze sau fișiere cu date. În particular, profilul de sincronizare este adresat încărcării de date pe un PDA sau notebook când părăsește locuința și colectarea de date de la acesta la revenire.

Era oare cu adevărat necesar să fie menționate toate aceste aplicații în detaliu și să fie create diferite stive de protocoale pentru fiecare dintre ele? Poate că nu era necesar, însă au existat diferite grupuri de lucru care au divizat standardul în părți mai mici și fiecare dintre ele și-a concentrat eforturile spre rezolvarea unei probleme specifice și a creat propriul profil. Gândiți-vă la aceasta ca la o aplicație pentru legea lui Conway (în aprilie 1968, în revista Datamation, Melvin Conway observa că atunci când propui la n persoane să scrie un compilator, primești ca soluție un compilator cu n-treceri de compilare, sau și mai general, structura software oglindește structura grupului care l-a produs). Era deci posibil să fie finalizate doar două stive de protocoale în loc de treisprezece, unul pentru transferul de fișiere și unul pentru fluidizarea comunicațiilor real-time.

4.6.3 Stiva de protocoale Bluetooth

Standardul Bluetooth conține mai multe protocoale grupate în niveluri. Structura nivelurilor nu respectă modelul OSI, modelul TCP/IP, modelul 802 sau oricare alt model existent. Totuși, IEEE lucrează la modificarea acestuia astfel încât să urmeze cât mai bine tiparul 802. Arhitectura de bază a protocolului Bluetooth, așa cum a fost modificată de comitetul 802, este prezentată în fig. 4-37.

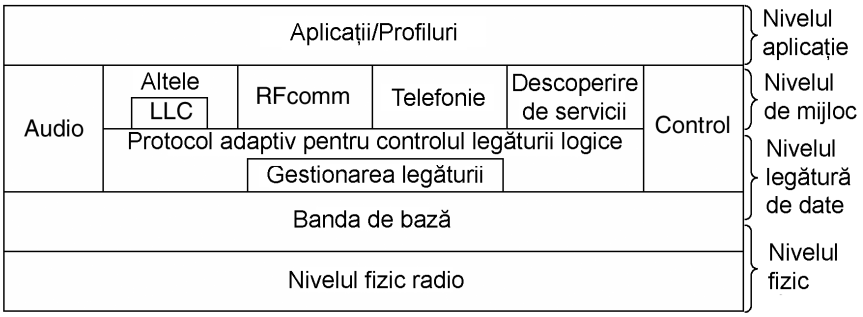


Fig. 4-37. Versiunea 802.15 a arhitecturii protocolului Bluetooth.

Nivelul de bază este reprezentat de nivelul fizic radio, care corespunde destul de bine nivelului fizic din modelele OSI și 802. El se ocupă cu transmisia radio și modularea semnalului. Atenția, în cadrul acestui nivel, s-a focalizat pe modelarea unui sistem ieftin astfel încât acesta să devină rapid un produs de larg consum. Nivelul bandă de bază este întrucâtva analog cu subnivelul MAC, însă include elemente de nivel fizic. El se ocupă de modul în care stăpânul controlează unitățile de timp și cum acestea sunt grupate în cadre.

Apoi urmează un nivel cuprinzând un grup de protocoale relativ asemănătoare. Gestionarul legăturii se ocupă cu stabilirea de canale logice între dispozitive, incluzând consumul de putere, autentificarea și calitatea serviciilor. Protocol adaptiv pentru controlul legăturii logice (adesea numit L2CAP – Logical Link Control Adaptation Protocol, rom: protocol de adaptare a controlului legăturii logice) oferă o interfață nivelurilor superioare prin ascunderea detaliilor de transmisie. El este analog cu subnivelul LLC din standardul 802, însă, din punct de vedere tehnic, este diferit de acesta. Așa cum sugerează și numele, protocolul de control și cel audio se ocupă de control și respectiv, de partea audio. Aplicațiile pot apela direct la acesta fără a avea nevoie de intermedierea protocolului L2CAP.

Nivelul următor este nivelul de mijloc și acesta conține o îmbinare de protocoale diferite. Nivelul LLC 802 a fost inserat aici de către IEEE pentru compatibilitatea sa cu celelalte rețelele 802. Protocoalele RFcomm, de telefonie și serviciul de descoperire sunt native. RFcomm (Radio Frequency communication, rom: comunicare pe frecvențe radio) este protocolul care emulează portul serial standard pe care îl găsim la orice PC obișnuit, pentru interconectarea de tastatură, mouse și modem, precum și pentru alte dispozitive. El a fost dezvoltat pentru a permite dispozitivelor perimate să îl folosească în continuare cu ușurință. Protocolul de telefonie este un protocol de timp real folosit pentru profilurile orientate pe 3 legături. El se ocupă și de stabilirea și de terminarea conexiunii. În cele din urmă, protocolul de descoperire a serviciilor este folosit pentru găsirea serviciilor din rețea.

Nivelul cel mai înalt este cel în care sunt localizate aplicațiile și profilurile. Ele se folosesc de protocoalele din nivelurile inferioare pentru a își realiza sarcinile. Fiecare aplicație are un subset propriu dintre aceste protocoale. Dispozitivele specifice, cum sunt căștile, conțin exclusiv acele protocoale necesare aplicației. În secțiunile următoare vom studia cele mai de jos trei niveluri din stiva de protocoale Bluetooth, deoarece acestea corespund în oarecare măsură cu subnivelurile fizic și MAC.

4.6.4 Nivelul Bluetooth radio

Nivelul radio transmite biții de la stăpân la sclav sau vice-versa. Este un sistem de putere mică cu o rază de acoperire de 10 metri, în banda de 2.4 GHz ISM. Banda este divizată în 79 de canale de 1

MHz fiecare. Modularea se face prin deplasarea în frecvență a cheilor, cu un bit pe Hz, realizând în total o rată de transfer de date de 1 Mbps, însă o mare parte din aceasta este consumată de supraîncărcare. Pentru a aloca rezonabil canalele, se folosește metoda FHSS (salturi de frecvență într-un spectru larg) cu 1600 salturi/sec și cu intervalul de timp de 62 μs. Toate nodurile dintr-un piconet rulează simultan, stăpânul dictând secvența de rulare.

Deoarece și 802.11 și Bluetooth operează în banda ISM de 2.4 GHz, pe aceleași 79 de canale, ele interferează unul cu celalalt. Întrucât rata de eșantionare a timpului Bluetooth este mult mai mare decât la 802.11, este mult mai probabil ca un dispozitiv Bluetooth să întrerupă transmisia unui 802.11 decât invers. Dat fiind faptul că 802.11 și 802.15 sunt amândouă standarde IEEE, organizația caută o soluție la această problemă, dar nu este deloc simplu știind că ambele sisteme folosesc banda ISM din același motiv: aceasta nu are nevoie de o licență de utilizare. Standardul 802.11a folosește cealaltă lungime de bandă ISM (5 GHz), însă are o rază mult mai scurtă decât cea a lui 802.11b (datorită fenomenelor fizice legate de undele radio), așadar folosirea 802.11a nu este o soluție foarte bună în toate cazurile. Câteva companii au rezolvat problema dând vina pe Bluetooth. O soluție de marketing ar fi ca rețeaua cu mai multă putere (politică și economică, nu electrică) să ceară părții mai slabe să își modifice standardul, pentru a nu mai interfera cu ea. Câteva idei în legătură cu această problemă sunt oferite în (Lansford et. co., 2001).

4.6.5 Nivelul bandă de bază Bluetooth

Nivelul bandă de bază este cea mai apropiată legătură pe care Bluetooth o are cu subnivelul MAC. Acesta transformă o secvența de biți într-un cadru și definește câteva formate de bază. În forma cea mai simplă, stăpânul din fiecare piconet definește o serie de cuante de timp de 625 μs; transmisia stăpânului se desfășoară în cuantele de timp pare, iar transmisia sclavilor în intervalele de timp impare. Aceasta este metoda tradițională de diviziune multiplexată a timpului, cu stăpânul preluând jumătate din perioadele de timp și sclavii împărțind cealaltă jumătate. Cadrele pot ocupa 1, 3 sau 5 cuante de timp.

În schema de alimentare sunt preconizate 250-260 μs pentru fiecare salt, pentru a putea permite stabilizarea circuitelor radio. Setări mai rapide decât atât sunt posibile numai la un cost mai ridicat. Pentru un singur cadru, după calibrare, 366 din cei 625 de biți sunt pierduți. Din aceștia, 126 sunt folosiți la codul de acces și la antet, lăsând ceilalți 240 de biți pentru date. Când cadrele ocupă cinci cuante de timp, este necesară o singură perioadă de stabilizare a circuitelor, așa că din 5 x 625 = 3125 de biți în cinci intervale de timp, 2781 sunt disponibili pentru nivelul bandă de bază. În concluzie, cadrele mai lungi sunt mult mai eficiente decât cele scurte, formate pe o singură cuantă de timp.

Fiecare cadru este transmis pe un canal logic, numit **legătură** (**link**), între stăpân și sclav. Există două tipuri de legături. Primul tip este denumit legătură **ACL** (**Asynchronous Connection-Less,** rom: Asincron fără conexiune) și este utilizat pentru comutarea de pachete de date ce sunt disponibile la intervale neregulate de timp. Aceste date sunt primite de la nivelul L2CAP la transmițător și sunt emise către nivelul L2CAP de la receptor. Traficul ACL este realizat în metoda „cea mai bună încercare" (best-effort). Nu sunt oferite nici un fel de garanții. Cadrele pot fi pierdute și atunci este necesară retransmiterea lor. Un sclav poate avea cel mult o legătură ACL la stăpânul său.

Celălalt tip este denumit legătură **SCO** (**Synchronous Connection Oriented,** rom: sincron orientat pe conexiune), fiind folosit pentru date reale, cum sunt transmisiile telefonice. Acestui tip de canal îi este alocat un număr fix de cuante de transmisie în fiecare direcție. Datorită naturii critice a legăturilor SCO, cadrele trimise pe aceste legături nu sunt niciodată retransmise. În schimb, pot fi

folosite mecanisme de corecție în avans a erorilor pentru a avea un grad mai mare de încredere. Un sclav poate avea până la trei legături SCO către stăpânul său. Fiecare legătură SCO poate transmite 64.000 bps PCM pe canal audio.

4.6.6 Nivelul L2CAP Bluetooth

Nivelul L2CAP are trei funcții majore. Prima funcție este aceea de a accepta pachete până la 64 KB de la nivelurile superioare și de a le sparge în cadre pentru transmisie. La sfârșit, cadrele sunt reasamblate în pachete.

A doua funcție este de a multiplexa și demultiplexa pachete provenite de la diverse surse. Când un pachet a fost reasamblat, nivelul L2CAP determină cărui protocol superior îi este adresat, spre exemplu RFcomm sau telefonic.

A treia funcție este aceea de a garanta calitatea serviciilor cerute, atât în timpul realizării conexiunii cât și în timpul operațiilor obișnuite. De asemenea, la configurare este negociat și maximul de informație utilă permisă, pentru a preveni situația în care un dispozitiv ce generează pachete mari suprasolicită un dispozitiv care folosește pachete mai mici. Această funcție este necesară deoarece nu toate dispozitivele pot utiliza pachete de 64 KB.

4.6.7 Structura cadrului Bluetooth

Există mai multe tipuri de cadre, cel mai important este reprezentat în fig. 4-38. El începe cu un cod de acces care identifică de obicei stăpânul; astfel, dacă un sclav se află în raza radio a doi stăpâni, va putea să știe cărui stăpân se adresează. Următorul câmp este un antet de 54 de biți, incluzând câmpurile tipice ale subnivelului MAC. Apoi urmează câmpul de date, de maxim 2744 de biți (pentru o transmisie de 5 intervale de timp). Pentru o singură unitate de timp formatul este același, cu excepția faptului că avem un câmp de date de 240 de biți.

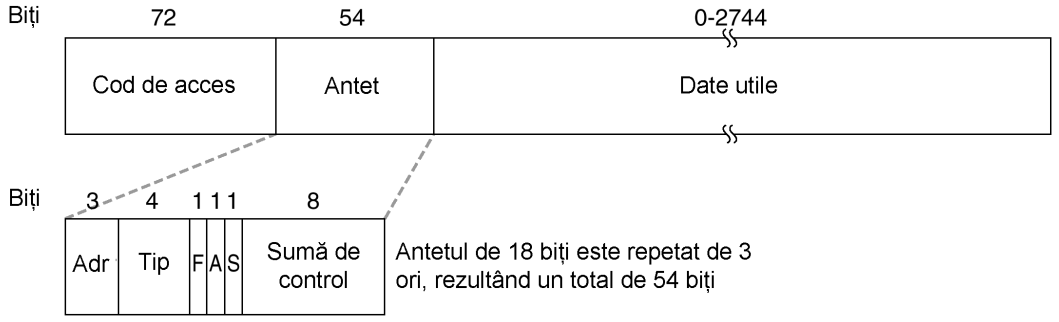


Fig. 4-38. Cadru de date Bluetooth tipic.

Să aruncăm acum o scurtă privire peste antet. Dintre cele opt dispozitive active, câmpul adresă îl identifică pe acela pentru care este destinat cadrul. Câmpul tip identifică tipul cadrului (ACL, SCO, interogare sau vid), tipul de corecție a erorii ce va fi folosit în câmpul de date și lungimea cadrului. Bitul de revărsare (flow) este folosit de către un sclav când memoria sa tampon este plină și, prin urmare, nu mai poate primi date. Bitul de confirmare pozitivă este folosit pentru a valida un cadru recepționat corect. Bitul de secvență este folosit pentru a numerota cadrele și este utilizat la retransmisii. Protocolul este de tipul pas-cu-pas, așa că un singur bit este suficient. Apoi urmează un câmp

de sumă de control de 8 biți. Tot antetul de 18 biți este repetat de trei ori pentru a forma antetul de 54 de biți din fig. 4-38. La receptor, un circuit simplu examinează cele trei copii ale fiecărui bit. Dacă toate sunt identice, bitul este acceptat. Dacă nu sunt identice, majoritatea va decide. Așadar 54 de biți din totalul de biți transmisibili sunt folosiți pentru a propaga antetul de 10 biți. Pentru că se dorește obținerea unei transmisii de date de încredere într-un mediu plin de interferențe, utilizând un dispozitiv ieftin, cu consum scăzut (2.5 mW) și cu capacitate redusă de calcul, se impune un nivel mare de redundanță în date.

Pentru câmpul de date din cadrele ACL sunt folosite mai multe forme. Totuși cadrele SCO sunt foarte simple: câmpul de date este mereu de 240 de biți. Trei variante sunt definite permițând o informație utilă de 80, 160 și 240 de biți, restul fiind utilizat pentru corecția de erori. În varianta cea mai de încredere (80 biți de informație utilă), conținutul este repetat de trei ori, la fel ca și antetul.

Deoarece un sclav nu poate folosi decât cuantele de timp impare, el primește 800 intervale de timp/sec, la fel ca și stăpânul. Cu o informație utilă de 80 de biți, capacitatea canalului de la sclav este 64000 bps, iar capacitatea canalului de la stăpân este tot de 64000 bps, suficient pentru un singur canal de voce PCM duplex-integral (acesta este motivul pentru care a fost aleasă o rată a salturilor de 1600 salturi/sec). Aceste date arată că un canal duplex-integral de voce cu 64.000 bps în ambele direcții, folosind cel mai de încredere format saturează complet un piconet, în ciuda unei benzi de transfer de 1 Mbps. Pentru varianta cu o încredere foarte mică (240 biți/interval de timp, fără redundanță la acest nivel), trei canale duplex-integral sunt suportate simultan, acesta fiind motivul pentru care un număr maxim de trei legături SCO sunt permise la un sclav.

Sunt multe lucruri de adăugat despre Bluetooth, însă din păcate nu avem un spațiu suficient aici. Pentru informații suplimentare, vă recomandăm (Bhagwat, 2001; Bisdikian, 2001; Bray și Sturman, 2002; Haartsen, 2000; Johansson și colab, 2001; Miller și Bisdikian, 2001; Sairam și colab., 2002).

4.7. COMUTAREA LA NIVELUL LEGĂTURII DE DATE

Multe organizații au mai multe LAN-uri și doresc să le conecteze. LAN-urile pot fi conectate prin dispozitive numite **punți** (**bridges**), care operează la nivelul legăturii de date. Punțile examinează adresele de la nivelul legăturii de date pentru a face rutarea. Întrucât ele nu trebuie sa examineze câmpurile cu informație utilă ale cadrelor pe care le rutează, ele pot transporta pachete IPv4 (utilizate acum în Internet), IPv6 (vor fi utilizate în Internet în viitor), AppleTalk, ATM, OSI, sau orice alt fel de pachete. Spre deosebire de punți, ruterele examinează adresele din pachete și fac rutarea pe baza acestora. Deși aceasta pare o departajare clară între punți și rutere, câteva îmbunătățiri de ultimă ora, precum apariția Ethernetului comutat, au complicat și mai mult lucrurile, cum vom vedea mai târziu. În capitolele care urmează ne vom ocupa de punți și comutatoare, în special pentru conectarea diferitelor LAN-uri 802. Pentru o tratare cuprinzătoare a punților, comutatoarelor și a altor subiecte înrudite, vezi (Perlman,1992).

Înainte de a intra în tehnologia punților, merită să aruncăm o privire asupra câtorva situații obișnuite în care acestea sunt folosite. Vom menționa șase motive pentru care o singură organizație poate ajunge să aibă LAN-uri multiple. În primul rând, multe universități și departamente ale unor corporații au propriile lor LAN-uri, în principal pentru a-și conecta calculatoarele personale, stațiile de lucru și serverele. Deoarece scopurile departamentelor diferă, departamente diferite vor alege

LAN-uri diferite, separat de ceea fac alte departamente. Mai devreme sau mai târziu, este nevoie de interacțiune, deci este nevoie de punți. În acest exemplu, LAN-urile multiple au apărut datorită autonomiei proprietarilor lor.

În al doilea rând, organizația poate fi răspândită geografic în mai multe clădiri separate aflate la distanțe considerabile. Poate fi mai ieftină soluția cu LAN-uri separate în fiecare clădire, conectate prin punți și legături în infraroșu, decât soluția cu întinderea unui singur cablu coaxial pe întreaga suprafață.

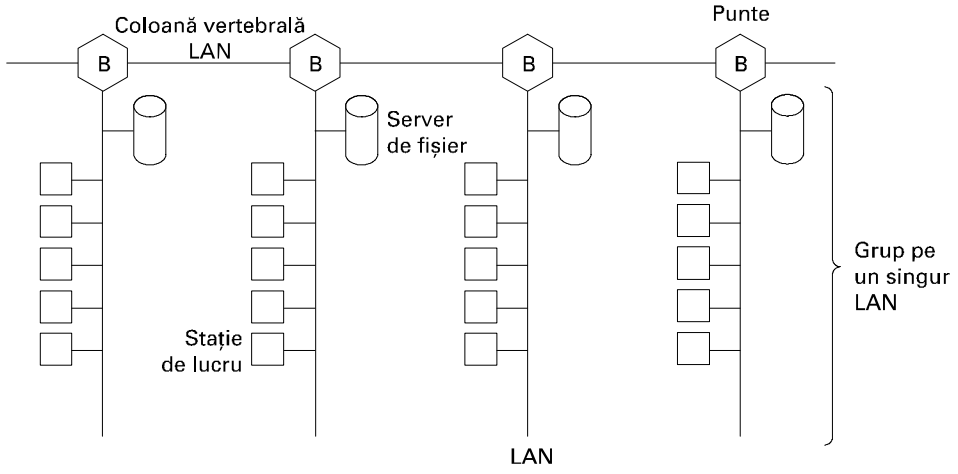


Fig. 4-39. LAN-uri multiple conectate printr-o coloană vertebrală pentru a trata un trafic total mai mare decât capacitatea unui singur LAN.

În al treilea rând, pentru a face față traficului, poate fi necesară spargerea unei entitați care din punct de vedere logic constituie un singur LAN în LAN-uri separate. La multe universități, de exemplu, mii de stații de lucru sunt disponibile pentru profesori și studenți. Fișierele sunt ținute de obicei pe servere și sunt încărcate pe mașinile utilizatorilor la cerere. Dimensiunea mare a acestui sistem împiedică punerea tuturor stațiilor de lucru pe un singur LAN - lărgimea de bandă totală necesară este mult prea mare. În schimb sunt folosite LAN-uri multiple conectate prin punți, după cum este arătat în fig. 4-39. Fiecare LAN conține un grup de stații de lucru cu propriul său server de fișiere, astfel încât cea mai mare parte a traficului este limitată la un singur LAN și astfel nu se încarcă suplimentar coloana vertebrală.

Merită amintit faptul că deși figurăm LAN-urile ca având acces la un același mediu de comunicație ca în fig. 4-39 (abordarea clasică), ele sunt cel mai frecvent implementate cu noduri sau, în zilele noastre, mai ales cu comutatoare. Oricum, un mediu de transmisie comun cu numeroase mașini conectate la el și un nod care conectează mașini sunt identice din punct de vedere funcțional. În ambele cazuri, toate mașinile aparțin aceluiași domeniu de coliziuni și toate utilizează protocolul CSMA/CD pentru a trimite cadre. Cum am văzut și mai înainte și cum vom vedea din nou în curând, LAN-urile comutate sunt diferite.

În al patrulea rând, există anumite situații în care un singur LAN ar fi potrivit în ceea ce privește traficul, dar distanța fizică între cele mai îndepărtate calculatoare este prea mare (de exemplu, mai mult de 2.5 km pentru Ethernet). Chiar dacă este ușor de întins cablul, rețeaua nu ar funcționa din cauza întârzierilor excesiv de mari pentru propagarea dus/întors a semnalelor. Singura soluție este

partiționarea LAN-ului și instalarea de punți între segmente. Folosind punțile, poate fi mărită distanța fizică totală acoperită.

În al cincilea rând, trebuie considerată problema siguranței. Pe un singur LAN, un nod defect, care trimite tot timpul un șir continuu de date alterate, va compromite LAN-ul. Punțile pot fi inserate în puncte critice pentru a preveni ca un singur nod care funcționează defectuos să afecteze întregul sistem. Spre deosebire de un repetor, care doar copiază ceea ce vede, o punte poate fi programată să exercite un anumit control privind ceea ce trimite mai departe și ceea ce nu trimite.

În al șaselea (și ultimul) rând, punțile pot contribui la securitatea organizației. Cele mai multe interfețe LAN au un mod **transparent de lucru** (promiscuous mode), în care *toate* cadrele sunt transferate calculatorului, nu numai cele care sunt adresate acestuia. Spionilor și băgăreților le place acest lucru. Prin inserarea punților în diferite locuri și prin grija de a nu transmite traficul de date sensibile, este posibilă izolarea unor părți din rețea, astfel încât datele să nu ajungă în mâinile cui nu trebuie.

Ideal ar fi ca punțile să fie perfect transparente, aceasta însemnând că ar fi posibilă mutarea unei mașini de pe un segment de cablu pe un altul fără a schimba nimic în hardware, în software, sau în tabelele de configurare. De asemenea, ar trebui să fie posibil ca o mașina de pe oricare segment să comunice cu mașini în oricare alt segment fără a ține seamă de tipul LAN-urilor folosite în cele două segmente sau în segmentele dintre ele. Acest țel este uneori atins, dar nu totdeauna.

4.7.1 Punți de la 802.x la 802.y

După ce am văzut de ce sunt necesare punțile, să ne întoarcem la felul în care funcționează acestea. Fig. 4-40 ilustrează funcționarea unei punți simple, dublu-port. Gazda A intr-un LAN fără fir (802.11) are un pachet de trimis către gazda fixă B într-un Ethernet (802.3) la care LAN-ul fără fir este conectat. Acest pachet coboară la subnivelul LLC și dobândește un antet LLC (figurat cu negru). Apoi trece la subnivelul MAC și îi este atașat un antet 802.11 (ca de altfel și o încheiere, care nu este figurată). Această structură este transmisă apoi prin aer și ajunge în cele din urmă la stația de la bază; aceasta observă că trebuie să trimită structura într-un LAN Ethernet. Apoi ajunge la puntea care conectează rețeaua 802.11 de rețeaua 802.3 la nivelul fizic și își continuă drumul spre nivelurile superioare.

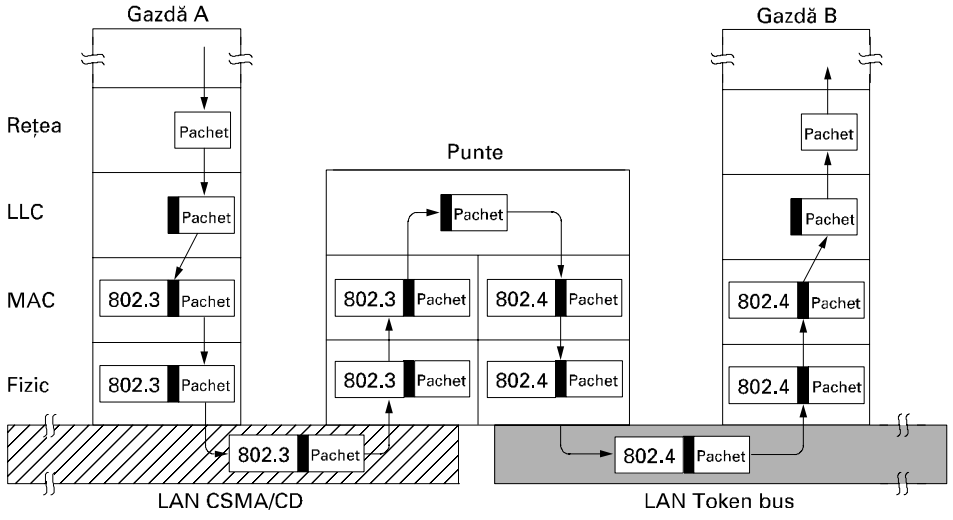


Fig. 4-40. Funcționarea unei punți de LAN de la 802.11 la 802.3.

În subnivelul MAC din punte, antetul 802.11 este îndepărtat. Pachetul simplu (cu antetul LLC) este predat subnivelului LLC din punte. În acest exemplu, pachetul este destinat unei subrețele 802.3 conectată la punte, astfel încât își face drum pe partea 802.3 a punții și pleacă mai departe în Ethernet. De notat că o punte conectând *k* LAN-uri diferite va avea *k* subniveluri MAC diferite și *k* niveluri fizice diferite, câte unul pentru fiecare tip.

Până acum, din câte am văzut, transmiterea unui pachet de la un LAN la altul pare simplă, însa nu este cazul. În această secțiune vom scoate în evidență câteva din dificultățile întâlnite atunci când se încearcă construirea unei punți între LAN-uri 802 diferite (și MAN-uri). Ne vom concentra atenția asupra 802.3, 802.11 și 802.16, dar mai sunt și altele cu seturile lor unice de probleme.

Pentru început, fiecare dintre LAN-uri folosește un format de cadru diferit (vezi fig. 4-41). Față de diferențele dintre Ethernet, token bus și token ring, care apăreau datorita egourilor marilor corporații și din cauza motivelor istorice, în acest caz, diferențele sunt bine argumentate. De exemplu, câmpul Durată la 802.11 este acolo datorită protocolului MACAW și nu are nici un sens în Ethernet. Prin urmare, orice transfer între LAN-uri diferite cere reformatare, ceea ce consumă timp de procesor, necesită o nouă calculare a sumei de control și introduce posibilitatea erorilor nedetectate datorată biților eronați în memoria punții.

O a două problemă este că LAN-urile interconectate nu funcționează neapărat la aceeași rată de transfer. Atunci când se transmite un șir lung de cadre concatenate de la un LAN rapid la unul mai lent, puntea nu va putea transmite cadrele în ritmul în care sosesc. De exemplu, dacă un gigabit Ethernet varsă biți într-un 11-Mb 802.11b LAN la viteză maximă, puntea va trebui să memoreze traficul, în speranța ca va avea memorie suficienta. Punțile care conectează trei sau mai multe LAN-uri au o problemă similară în cazul în care mai multe LAN-uri încearcă să alimenteze același LAN de ieșire în același moment chiar dacă toate LAN-urile au aceeași viteză.

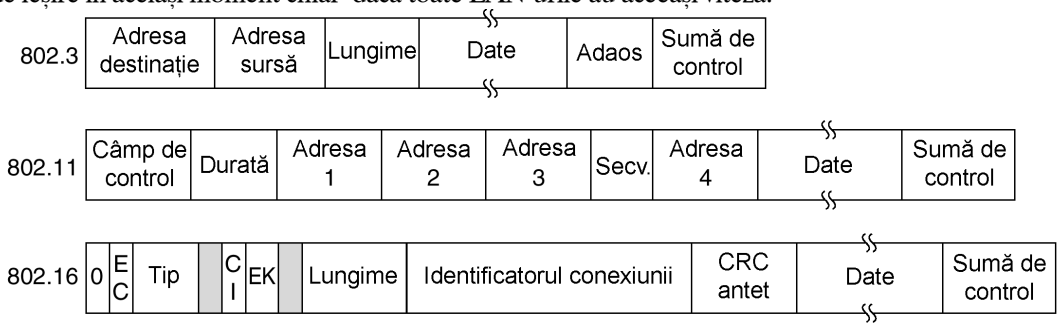


Fig. 4-41. Formatele cadrelor IEEE 802. Desenul nu este la scară.

O a treia, și potențial cea mai serioasă problemă dintre toate, este că diferite LAN-uri 802 au o lungime maximă de cadru diferită. O problemă evidentă apare atunci când un cadru lung trebuie transmis unui LAN care nu îl poate accepta. La acest nivel, împărțirea cadrului iese din discuție. Toate protocoalele presupun recepționarea totală sau deloc a cadrelor. Nu există posibilitatea de reasamblare a cadrelor din unități mai mici. Aceasta nu înseamnă că asemenea protocoale nu ar putea fi inventate. Ele pot fi și au fost. Doar că nici un protocol legătură de date nu are această caracteristică, așa că punțile nu trebuie să se atingă de informația utilă din cadru. Fundamental, nu există nici o soluție. Cadrele care sunt prea lungi pentru a fi transmise trebuie eliminate. Cam atât în ceea ce privește transparența.

Un alt punct este securitatea. 802.11 și 802.16 suportă criptarea la nivelul legăturii de date. Ethernet nu suportă. Aceasta înseamnă că diversele servicii de criptare disponibile la rețelele fără fir sunt pierdute când traficul trece printr-o rețea Ethernet. Încă și mai rău, dacă o stație fără fir utili-

zează criptare la nivelul legăturii de date, nu există nici o modalitate de a decripta datele când ajung în Ethernet. Dacă o stație fără fir nu utilizează criptarea, traficul acesteia este expus de-a lungul legăturii prin aer. În ambele situații există o problemă.

O soluție la problema securității ar fi criptarea la un nivel superior, dar în acest fel o stație 802.11 trebuie să afle dacă vorbește cu o altă stație într-o rețea 802.11 (însemnând că folosește criptare la nivelul legăturii de date) sau nu (însemnând că nu folosește). Forțarea unei stații să facă o alegere distruge transparența.

Punctul final este calitatea serviciilor. Atât 802.11 cât și 802.16 o oferă în forme diverse, primul folosind modul PCF și ultimul folosind conexiuni cu rată de transfer constantă. Ethernet-ul nu oferă nimic în acest sens, așa că traficul de la oricare dintre ceilalți va pierde din calitate atunci când trece printr-o rețea Ethernet.

4.7.2 Interconectarea locală a rețelelor

Capitolul anterior s-a ocupat de problemele întâmpinate la conectarea printr-o singură punte a două LAN-uri IEEE 802 diferite. Oricum, în organizațiile mari cu multe LAN-uri, numai simpla interconectare a acestora ridică mai multe probleme, chiar și dacă toate LAN-urile sunt Ethernet. Ideal, ar trebui să fie posibil să te duci și să cumperi punți proiectate după standardul IEEE, să le conectezi și totul să funcționeze perfect, instantaneu. Nu ar trebui să fie nevoie de modificări de hardware, de modificări de software, de setarea adreselor, de încărcarea tabelelor sau parametrilor, de nimic altceva. Se conectează numai cablurile și funcționează. Mai mult, funcționarea LAN-urilor existente nu ar trebui să fie afectată în nici un fel de punți. Cu alte cuvinte punțile ar trebui să fie complet transparente (invizibile pentru hardware și software). Destul de surprinzător, chiar au reușit. Haideți să vedem cum se realizează această magie.

O punte transparentă operează în mod transparent (promiscuous mode), acceptând orice cadru transmis pe oricare dintre LAN-urile la care este atașată. De exemplu, să considerăm configurația din fig. 4-42. Puntea B1 este conectată la LAN-urile 1 și 2, iar puntea B2 este conectată la LAN-urile 2, 3 și 4. Un cadru destinat lui A de la LAN 1 care ajunge la puntea B1 poate fi eliminat imediat, pentru că este deja pe LAN-ul care trebuie, dar un cadru care ajunge de la LAN 1 pentru C sau F trebuie transmis.

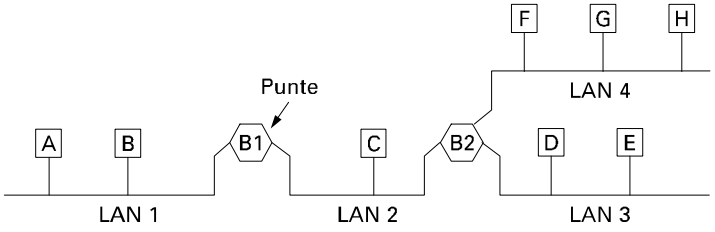


Fig. 4-42. O configurație cu patru LAN-uri și două punți.

La sosirea unui cadru, o punte trebuie să decidă dacă să îl elimine sau să îl transmită mai departe, iar dacă îl transmite, către ce LAN să îl trimită. Această decizie este luată căutând adresa destinației într-o tabelă de dispersie menținută în interiorul punții. Tabelul poate să includă fiecare destinație posibilă și cărei linii de ieșire (de fapt, cărui LAN) îi aparține. De exemplu, tabelul lui B2 ar include A ca aparținând lui LAN 2, din moment ce tot ce trebuie să știe B2 este către care LAN să trimită cadrele destinate lui A. Nu prezintă interes faptul că ulterior vor avea loc mai multe transmisii.

La prima conectare a punților, toate tabelele de dispersie sunt vide. Nici una dintre punți nu știe unde se află destinațiile, astfel încât toate folosesc algoritmul de inundare: orice cadru care vine pentru o destinație necunoscută este trimis către toate LAN-urile la care este conectată puntea, cu excepția celui din care a venit. Cu trecerea timpului, punțile află unde se găsesc destinațiile, după cum este descris în cele ce urmează. Odată ce o destinație este cunoscută, cadrele destinate ei sunt puse pe LAN-ul care trebuie, în loc să fie inundate.

Algoritmul folosit de punțile transparente se numește *învățare regresivă* (*backward learning*). După cum a fost menționat anterior, punțile lucrează în mod transparent (promiscuous), astfel încât toate văd fiecare cadru trimis pe oricare dintre LAN-urile lor. Uitându-se la adresa sursei, ele pot afla care calculator este accesibil pe care LAN. De exemplu, dacă puntea B1 din fig. 4-38 vede un cadru din LAN 2 venind de la C, știe că stația C trebuie să fie accesibilă prin LAN 2 și creează o intrare în tabela de dispersie, în care notează că pentru cadrele care merg la C ar trebui să folosească LAN 2. Orice cadru ulterior adresat lui C care vine din LAN 1 va fi transmis mai departe, pe când un cadru pentru C venit din LAN 2 va fi abandonat.

Topologia se poate schimba după cum calculatoarele și punțile sunt în funcțiune sau nu, sau mutate de colo-colo. Pentru a trata topologii dinamice, de câte ori se creează o intrare în tabela de dispersie, în ea este notat timpul de sosire a cadrului. De câte ori sosește un cadru a cărui destinație se află deja în tabel, intrarea sa este adusă la zi cu timpul curent. Astfel, timpul asociat fiecărei intrări arată ultimul moment în care a fost primit un cadru de la respectivul calculator.

Periodic, un proces din punte scanează tabela de dispersie și curăță toate intrările mai vechi de câteva minute. În acest fel, dacă un calculator este scos din LAN-ul său, plimbat prin clădire și reinstalat în altă parte, în câteva minute va reveni la funcționarea normală, fără vreo intervenție manuală. Acest algoritm semnifică de asemenea că dacă un calculator este inactiv pentru câteva minute, orice trafic trimis spre el va trebui inundat, până când calculatorul respectiv va trimite un cadru.

Procedura de dirijare pentru un cadru sosit depinde de LAN-ul din care sosește (LAN-ul sursă) și de LAN-ul în care se află destinația sa (LAN-ul destinație), după cum urmează:

1. Dacă LAN-ul sursă este același cu LAN-ul destinație, abandonează cadrul.
2. Dacă LAN-ul sursă și cel destinație sunt diferite, transmite cadrul.
3. Dacă LAN-ul destinație nu este cunoscut, folosește inundarea.

Acest algoritm trebuie aplicat pentru fiecare cadru care sosește. Există cipuri VLSI speciale care realizează căutarea și actualizarea în tabela de dispersie, doar în câteva microsecunde.

4.7.3 Punți cu arbore de acoperire

Pentru a mări siguranța, unele locații folosesc două sau mai multe punți în paralel între perechi de LAN-uri, așa cum este arătat în fig. 4-43. Totuși, acest aranjament introduce și unele probleme suplimentare, întrucât creează bucle în topologie.

Un simplu exemplu al acestor probleme poate fi văzut în fig. 4-43, observând modul în care este tratat cadrul F cu destinație necunoscută. Fiecare punte, urmând regulile obișnuite pentru tratarea destinațiilor necunoscute, folosește inundarea care, în acest exemplu, nu înseamnă decât copierea cadrului pe LAN 2. Puțin după aceea, puntea 1 vede F2, un cadru cu destinație necunoscută, pe care îl copiază pe LAN 1, generând F3 (care nu este arătat în figură). La fel, puntea 2 copiază F1 pe LAN1 generând F4 (care nu este arătat). Acum puntea 1 trimite F4 și puntea 2 copiază F3. Acest ciclu se continuă la nesfârșit.

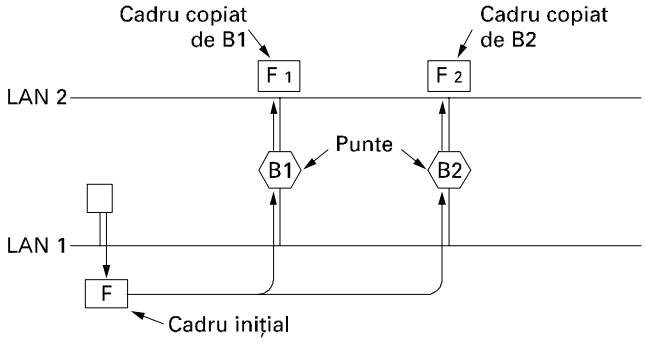


Fig. 4-43. Două punți transparente paralele.

Soluția acestei probleme este ca punțile să comunice unele cu altele și să suprapună peste topologia actuală un arbore de acoperire care ajunge la fiecare LAN. De fapt, în interesul construirii unei topologii fictive fără bucle, sunt ignorate câteva conexiuni posibile între LAN-uri. De exemplu, în fig. 4-44(a) apar nouă LAN-uri interconectate prin zece punți. Această configurație poate fi rezumată într-un graf cu LAN-urile drept noduri. Un arc leagă oricare două LAN-uri care sunt conectate de o punte.

Graful poate fi redus la un arbore de acoperire renunțând la arcurile figurate ca linii punctate în fig. 4-44(b). Folosind acest arbore de acoperire, există un singur drum de la fiecare LAN la fiecare alt LAN. Odată ce punțile s-au înțeles asupra arborelui de acoperire, toată transmiterea dintre LAN-uri urmărește arborele de acoperire. Din moment ce există un drum unic de la fiecare sursă la fiecare destinație, buclele sunt imposibile.

Pentru a construi arborele de acoperire, punțile trebuie să aleagă mai întâi o punte care va reprezenta rădăcina arborelui. Ele fac această alegere prin emiterea de către fiecare punte a numărului de serie, instalat de fabricant, garantat ca fiind unic în întreaga lume. Puntea cu cel mai mic număr serial devine rădăcină. Apoi se construiește un arbore de drumuri minime de la rădăcină la fiecare punte și LAN. Acest arbore este un arbore de acoperire. Dacă o punte sau un LAN cade, trebuie calculat un nou arbore de acoperire.

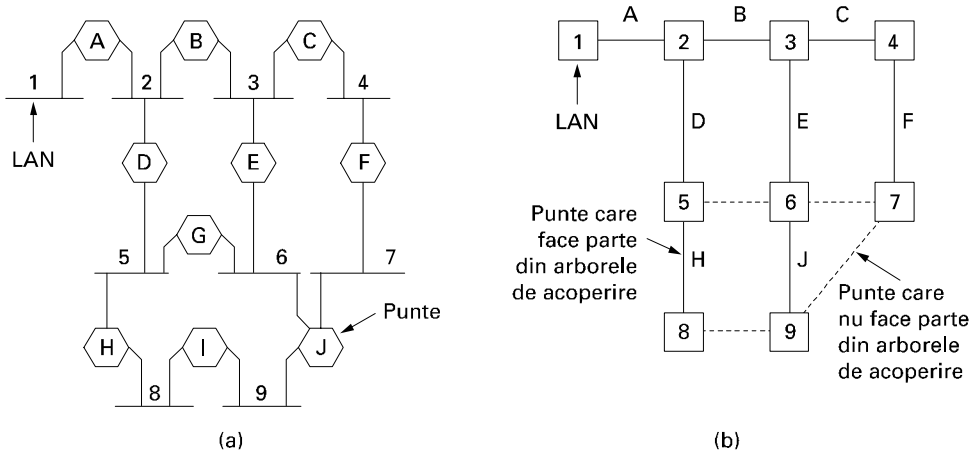


Fig. 4-44. (a) LAN-uri interconectate. (b) Arbore de acoperire pentru LAN-uri. Liniile punctate nu fac parte din arborele de acoperire.

Rezultatul acestui algoritm este că se stabilește un drum unic de la fiecare LAN la rădăcină, și astfel la fiecare alt LAN. Deși arborele acoperă toate LAN-urile, nu neapărat toate punțile sunt prezente în arbore (pentru a evita buclele). După ce a fost stabilit arborele de acoperire, algoritmul continuă să ruleze pentru a detecta automat schimbări în topologie și a actualiza arborele. Algoritmul distribuit, folosit pentru construirea arborelui de acoperire, a fost inventat de Perlman și este descris în detaliu în (Perlman, 1992). Acesta este standardizat în IEEE 802.1D.

4.7.4 Punți aflate la distanță

Punțile sunt de cele mai multe ori folosite pentru conectarea a două (sau mai multe) LAN-uri aflate la distanță unele de altele. De exemplu, o companie poate avea fabrici în mai multe orașe, fiecare dintre acestea cu propriul său LAN. Ideal ar fi ca toate aceste LAN-uri să fie interconectate pentru ca sistemul în întregime să funcționeze ca un mare LAN.

Acest țel poate fi atins punând câte o punte fiecărui LAN și conectând punțile în perechi cu linii punct-la-punct (de exemplu linii închiriate de la o companie de telefoane). Un sistem simplu, cu trei LAN-uri, este prezentat în fig. 4-45. Aici se aplică algoritmul de dirijare obișnuit. Cel mai simplu este să se privească cele trei linii punct-la-punct ca LAN-uri fără gazde. Adică, un sistem obișnuit de șase LAN-uri interconectate prin patru punți. Nimic din ce am studiat până acum nu spune că un LAN trebuie să aibă gazde.

Pe liniile punct-la-punct pot fi folosite diverse protocoale. O posibilitate este alegerea unui protocol de legătură de date punct-la-punct standard, cum ar fi PPP, punând cadre MAC complete în câmpul de informație utilă. Această strategie funcționează cel mai bine dacă LAN-urile sunt identice și singura problemă este transmiterea cadrelor la LAN-ul care trebuie. Altă posibilitate este eliminarea antetului și a încheierii cadrelor MAC la puntea sursă, punând ceea ce a mai rămas în câmpul de informație utilă al protocolului punct-la-punct. Un nou antet și o nouă încheiere MAC pot fi apoi generate la puntea destinație. Un dezavantaj al acestei abordări este că suma de control care ajunge la puntea destinație nu este cea calculată de gazda sursă, existând posibilitatea ca erori cauzate de biți eronați în memoria unei punți să nu fie detectați.

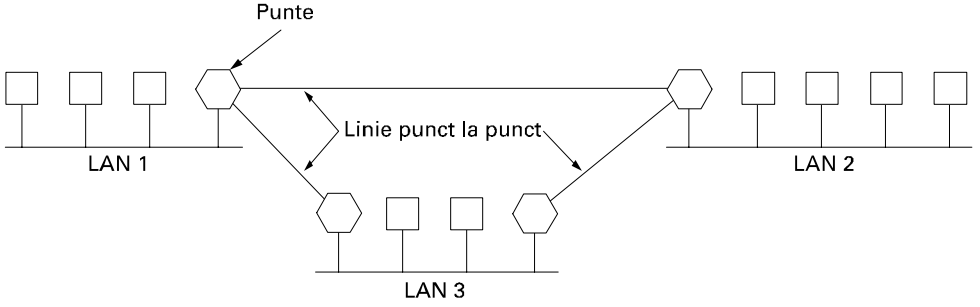


Fig. 4-45. Punți aflate la distanță folosite pentru a interconecta LAN-uri îndepărtate.

4.7.5 Repetoare, Noduri, Punți, Comutatoare, Rutere și Porți

Până acum în aceasta carte am văzut a mulțime de feluri de a transfera cadre și pachete de pe un segment de cablu pe altul. Am amintit de repetoare, noduri, punți, comutatoare, rutere și porți. Toa-

te aceste dispozitive sunt utilizate în mod curent, dar ele diferă mai mult sau mai puțin unul de altul. Deoarece sunt atât de multe, merită sa le analizăm împreună pentru a vedea asemănările și diferențele dintre ele.

Pentru început, aceste dispozitive operează la niveluri diferite, cum este ilustrat în fig. 4-46(a). Nivelul contează pentru că diferitele dispozitivele folosesc segmente diverse din informație pentru a decide cum să comute. Într-un scenariu tipic, utilizatorul creează date pentru a fi trimise către o mașina aflată la distanță. Aceste date sunt trimise nivelului transport, unde li se adaugă un antet, de exemplu un antet TCP, și se transmite rezultatul mai jos către nivelul rețea. Nivelul rețea adaugă propriul antet pentru a forma un pachet pentru nivelul rețea, de exemplu un pachet IP. În fig. 4-46(b) observăm pachetul IP colorat in gri. Apoi pachetul ajunge la nivelul legăturii de date, care îi adaugă propriul antet și suma de control (CRC) și trimite cadrul rezultat către nivelul fizic pentru transmisie, de exemplu intr-un LAN.

Acum să ne uităm la dispozitivele de comutare și să vedem legătura lor cu pachetele și cadrele. La cel mai de jos nivel, nivelul fizic, se află repetoarele. Acestea sunt dispozitive analogice ce sunt conectate intre două segmente de cablu. Un semnal ce apare pe unul din aceste cabluri este amplificat și trimis pe celălalt cablu. Repetoarele nu înțeleg cadrele, pachetele sau antetele. Ele înțeleg doar tensiuni electrice. Ethernetul clasic, de exemplu, a fost proiectat sa permită folosirea a patru repetoare în scopul de a extinde lungimea maximă a cablului de la 500 de metri la 2500 de metri.

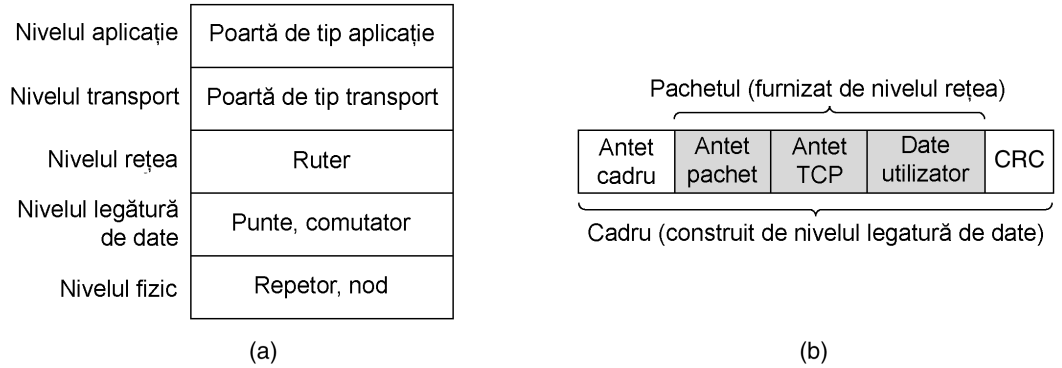


Fig. 4-46. (a) Corespondența dintre niveluri și dispozitive. (b) Cadre, pachete și antete

În continuare ajungem la noduri. Un nod are un număr de linii de intrare pe care le unește din punct de vedere electric. Cadrele care ajung la nod pe oricare linie sunt trimise afară pe toate celelalte liniile. Dacă două cadre ajung în același timp se vor ciocni la fel ca și atunci când ar fi transmise pe un cablul coaxial. Cu alte cuvinte un nod formează un singur domeniu de coliziune. Toate liniile ce intră în nod trebuie să lucreze la aceeași viteză. Nodurile diferă de repetoare prin faptul ca (de obicei) nu amplifică semnalele pe care le primesc și sunt proiectate pentru a suporta multe plăci de extensie cu mai multe intrări; totuși, diferențele nu sunt semnificative. Ca și repetoarele, nodurile nu examinează adresele 802 și nici nu le utilizează în vreun fel. Un nod este arătat în fig. 4-47(a).

Acum vom aborda nivelul legăturii de date, unde găsim punțile și comutatoarele. Tocmai am studiat punțile. O punte conectează două sau mai multe LAN-uri așa cum este arătat în fig. 4-47(b). Când un cadru ajunge, software-ul din punte extrage adresa destinație din cadru și caută în tabela sa vadă unde să trimită cadrul. Pentru Ethernet, această adresă este adresa destinație de 48 de biți prezentată în fig. 4-17. Asemănător unui nod, o punte moderna are placi de extensie, de obicei pentru patru sau opt intrări de un anumit tip. O placă de extensie pentru Ethernet nu poate manevra, sa

zicem, cadre token ring, pentru că nu știe unde să găsească adresa destinație în antetul cadrului. Oricum, o punte poate avea placi de extensie pentru diferite tipuri de rețele și diferite viteze. Spre deosebire de nod, la punte fiecare linie se află în propriul domeniu de coliziune.

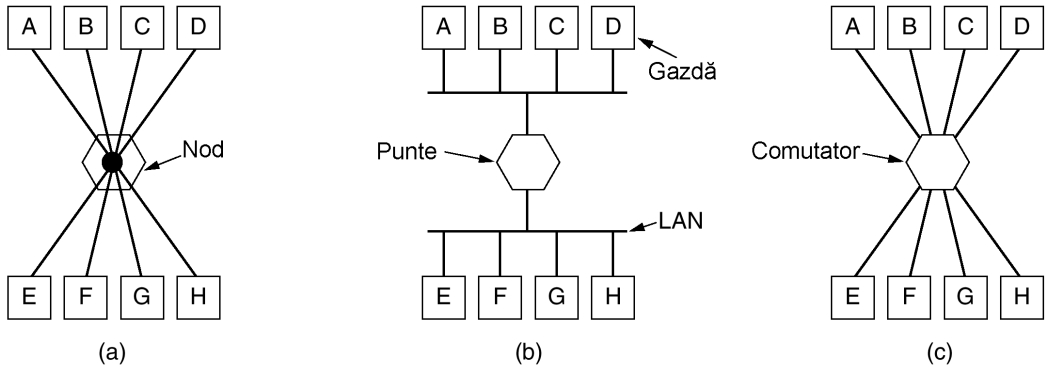


Fig. 4-47. (a) Un nod. (b) O punte. (c) Un comutator.

Comutatoarele sunt similare cu punțile deoarece amândouă rutează cadre pe baza adreselor. De fapt, mulți oameni folosesc aceste două denumiri fără a face o distincție clară intre termeni. Principala diferență este aceea că un comutator este cel mai adesea folosit pentru a conecta calculatoare individuale, așa cum este arătat în fig. 4-47(c). Ca o consecință, când gazda A în fig. 4-47(b) dorește sa trimită un cadru către gazda B, puntea primește cadrul dar nu îl ia în considerare. Din contră, după cum se vede în fig. 4-47(c), comutatorul trebuie sa retransmită cadrul de la A la B deoarece nu există alt drum pentru ca acest cadru să ajungă. Întrucât fiecare port al comutatorului este de obicei conectat la un singur calculator, comutatorul trebuie să aibă loc pentru mai multe placi de extensie decât punțile care trebuie să conecteze numai rețele. Fiecare placă de extensie are un spațiu tampon pentru cadrele recepționate. Deoarece fiecare port se află în propriul domeniu de coliziune, comutatoarele nu pierd niciodată cadre din cauza coliziunilor. Totuși, dacă un comutator primește cadre mai repede decât le poate retransmite, este posibil ca în scurt timp sa nu mai aibă memorie tampon liberă și sa înceapă să arunce din cadrele primite.

Pentru a relaxa puțin problema, comutatoarele moderne încep sa retransmită cadre imediat ce antetul destinație ajunge, dar înainte ca restul cadrului sa ajungă (bineînțeles asigurându-se ca linia de ieșire este disponibila). Aceste comutatoare nu utilizează tehnica de memorare și retransmitere. Câteodată ele sunt menționate drept comutatoare cu transmitere de fragmente (cut-through switches). De obicei acest tip de comutator este implementat în întregime în hardware, în timp ce tradiționalele punți conțin un CPU ce face comutare cu memorare și retransmitere la nivel software. Dar deoarece toate punțile și comutatoarele moderne conțin circuite integrate speciale pentru comutare, diferențele dintre comutatoare și punți țin mai mult de probleme de marketing decât de probleme tehnice.

Până acum, am văzut repetoare și noduri, care sunt foarte asemănătoare, precum și punți și comutatoare, care sunt de asemenea foarte asemănătoare între ele. Acum trecem mai departe la rutere, care sunt și ele diferite de cele menționate mai sus. Când un pachet ajunge la un ruter, antetul și sfârșitul cadrului sunt eliminate și pachetul localizat în informația utila a cadrului (înnegrit în fig. 4-46) trece către software-ul de rutare. Acest software folosește antetul pachetului pentru a alege o linie de ieșire. Pentru un pachet IP, antetul pachetului va conține adrese de 32 de biți (IPv4) sau

adrese de 128 de biți (IPv6), în nici un caz adrese 802 de 48 de biți. Software-ul de rutare nu vede adresele cadrelor și nici măcar nu știe daca pachetul a venit de pe un LAN sau de pe o linie punct la punct. În cap. 5 vom studia ruterele și rutarea.

Mai sus cu un nivel găsim porțile de transport (gateways). Acestea conectează doua calculatoare ce utilizează diferite protocoale de transport orientate pe conexiune. De exemplu, să presupunem că un calculator care utilizează protocolul TCP/IP orientat pe conexiune, trebuie să discute cu un calculator care folosește protocolul ATM orientat pe conexiune. Poarta de transport poate copia pachete de la o conexiune la alta, refăcând pachetele după necesitați.

În încheiere, porțile la nivelul aplicație înțeleg formatul și conținutul datelor și traduc mesajul de la un format la altul. De exemplu, o poartă de poșta electronică poate traduce mesaje Internet în mesaje SMS pentru telefoane mobile.

4.7.6 LAN-uri virtuale

La începutul dezvoltării rețelelor locale de calculatoare, cabluri groase galbene șerpuiau prin conductele de cablu ale multor clădiri de birouri. Acestea conectau toate calculatoarele pe la care treceau. Adesea erau mai multe cabluri conectate la coloana vertebrală centrală (ca în fig. 4-39) sau la un nod central. Nu se acorda nici o importanță corespondenței între calculatoare și LAN-uri. Toți oamenii din birouri alăturate erau conectați la același LAN idiferent dacă aparțineau sau nu aceleiași organizații. Poziționarea fizică a calculatoarelor domina logica.

Totul s-a schimbat odată cu apariția lui 10Base-T și a nodurilor în anii 1990. Clădirile au fost recablate (cu costuri considerabile) pentru a scoate vechile cabluri galbene, instalându-se în loc cabluri cu perechi de fire torsadate de la fiecare birou până la locurile de conectare centrală de la capătul fiecărui coridor sau până la camera unde se afla calculatorul principal, așa cum este ilustrat în fig. 4-48.

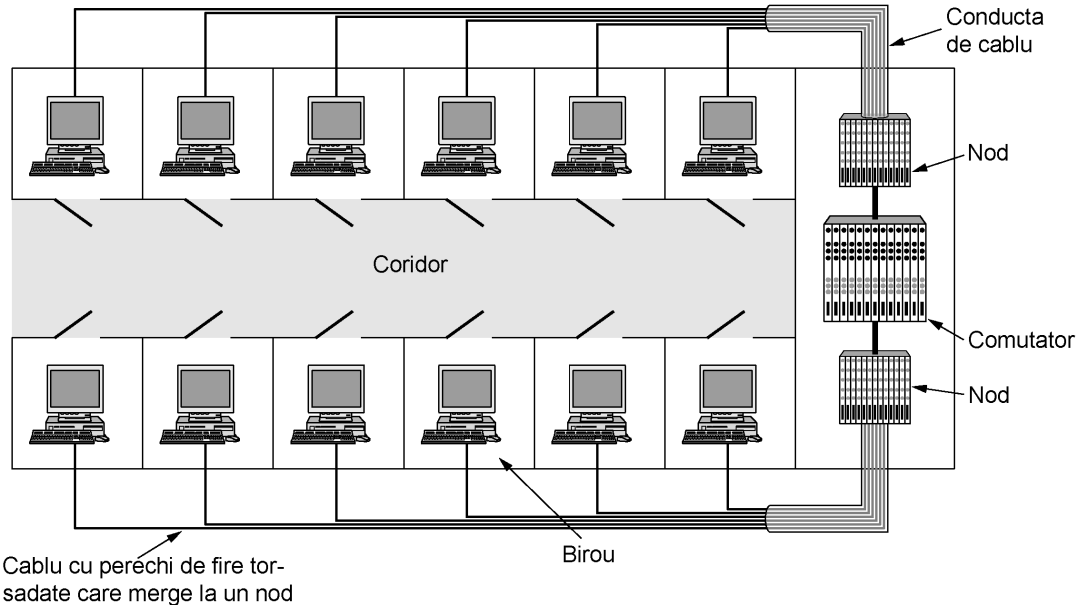


Fig. 4-48. O clădire cu rețea centralizata ce folosește noduri și un comutator.

Dacă vicepreședintele care se ocupa de cablare era vizionar, se instalau cabluri cu perechi de fire torsadate de categoria a cincea; dacă acesta era un statistician era utilizat cablu de telefon existent (categoria 3) (pentru a fi înlocuit câțiva ani mai târziu când a apărut Ethernet-ul rapid).

Folosindu-se noduri (și mai târziu, comutatoare) Ethernet, era adesea posibilă configurarea LAN-urilor din punct de vedere logic mai mult decât din punct de vedere fizic. Daca o companie dorește k LAN-uri, cumpără k noduri. Alegând cu grijă ce conectoare vor fi introduse în nod, ocupanții LAN-ului pot fi aleși din punct de vedere organizațional fără a se da prea mare importanță amplasării geografice. Bineînțeles, dacă doi oameni aparținând aceluiași departament lucrează în clădiri diferite, cel mai probabil aceștia sunt conectați în noduri diferite și astfel în LAN-uri diferite. Cu toate acestea, situația se prezintă mult mai bine decât în cazul LAN-urilor bazate pe amplasare geografică.

Contează cine este conectat și în ce LAN? Oricum, în toate organizațiile, toate LAN-urile sunt în cele din urmă interconectate. Pe scurt, răspunzând la întrebare: adesea contează. Dintr-o multitudine de motive administratorii de rețea își doresc să grupeze utilizatorii în LAN-uri pentru a reflecta structura organizatorică mai degrabă decât structura fizică a clădirii. O problemă este securitatea. Orice interfață de rețea poate fi configurată în mod transparent, copiind tot traficul care sosește pe canalul de comunicație. Multe departamente, cum ar fi cele de cercetare, patente și contabilitate, dețin informații pe care nu doresc să le facă cunoscute în afara departamentului. În situații ca acestea, punerea tuturor oamenilor din departament într-o singură rețea locală fără a permite vreunui fel de trafic să iasă din această rețea este o soluție bună. Totuși, administratorul va vrea sa audă despre astfel de aranjamente doar dacă toți oamenii din fiecare departament sunt localizați in birouri adiacente, fără birouri interpuse între acestea.

Poate să apară și o a doua problemă. Unele rețele locale sunt utilizate mai intensiv decât altele și poate fi benefică separarea acestora la anumite momente. De exemplu, dacă persoanele de la cercetare rulează tot felul de experimente dichisite care din când în când scapă de sub control și le saturează rețeaua locală, persoanele de la contabilitate s-ar putea să nu fie foarte entuziasmate în a dona capacitatea departamentului lor pentru a ajuta.

O a treia problemă este difuzarea. Cele mai multe rețele locale suportă difuzarea și multe protocoale de nivel superior folosesc această facilitate în mod extensiv. Spre exemplu, atunci când un utilizator dorește să trimită un pachet către o adresă IP x, cum știe stația sa ce adresă MAC să pună in cadru? Vom studia această întrebare în cap. 5, dar, pe scurt, răspunsul este că va difuza un cadru care conține întrebarea: A cui este adresa IP x? Apoi așteaptă un răspuns. Și există multe alte exemple de utilizare a difuzării. Pe măsură ce din ce în ce mai multe rețele locale sunt interconectate, numărul cadrelor de difuzare recepționate de fiecare mașină tinde să crească liniar cu numărul de mașini.

O altă problemă legată de difuzare apare din când în când, atunci când o placă de rețea se defectează și începe să transmită un șir nesfârșit de cadre de difuzare. Rezultatul acestei **furtuni de difuzări** (broadcast storm) este că (1) întreaga capacitate a rețelei locale este ocupată de aceste cadre și (2) că toate mașinile din toate rețele locale interconectate cu aceasta sunt paralizate doar prin procesarea și ignorarea tuturor cadrelor difuzate.

La prima vedere s-ar putea părea că furtunile de difuzări pot fi limitate în spațiu prin separarea rețelelor locale prin punți și comutatoare, dar dacă scopul este să se atingă transparență (de ex o mașină poate fi mutată intr-o rețea locală diferită fără ca nimeni să observe acest lucru), atunci punțile trebuie să înainteze toate cadrele de difuzare.

După ce am văzut de ce companiile ar dori să aibă mai multe rețele locale cu întindere limitată, haideți să ne întoarcem la problema decuplării topologiei logice de cea fizică. Să presupunem că un utilizator este mutat în cadrul companiei de la un departament la altul fără să își schimbe biroul, sau

că își schimbă biroul fără a-și schimba departamentul. Folosind o cablare bazată pe hub-uri, mutarea utilizatorului în rețeaua locală corectă presupune ca administratorul de rețea să meargă în centrul de cablare și să mute conectorul pentru calculatorul utilizatorului respectiv dintr-un hub în alt hub.

În multe companii, schimbările organizaționale au loc tot timpul, însemnând că administratorii de sistem petrec o mulțime de timp scoțând cabluri de undeva și punându-le în altă parte. De asemenea, în unele cazuri, este posibil ca schimbările să nu poată fi făcute deloc, pentru că perechea torsadată de la mașina utilizatorului este prea departe de hub-ul potrivit (de ex. în altă clădire).

Ca răspuns la cerințele utilizatorilor pentru o flexibilitate sporită, comercianții de echipamente de rețea au început să lucreze la o modalitate de a recabla clădiri în întregime doar cu ajutorul software-ului.. Conceptul rezultat este numit **VLAN (Virtual LAN**, rom: rețea locală virtuală) și a fost standardizat de către comitetul 802. Acum este utilizat în multe organizații. Haideți să aruncăm o privire asupra lui. Pentru informații suplimentare despre VLAN-uri, vezi (Breyer and Riley, 1999; and Seifert, 2000).

VLAN-urile se bazează pe comutatoare dedicate, cu toate că pot avea niște hub-uri la periferie, ca în fig. 4-48. Pentru configurarea unei rețele bazate pe VLAN-uri, administratorul de rețea decide câte VLAN-uri vor exista, ce calculatoare vor aparține fiecărui VLAN și cum se vor numi VLAN-urile. De cele mai multe ori, VLAN-urile sunt denumite (informal) cu nume de culori, pentru că este apoi posibilă tipărirea de diagrame color cu dispunerea fizică a mașinilor, figurând membrii VLAN-ului roșu în roșu, membrii VLAN-ului verde în verde și așa mai departe. În acest fel, atât dispunerea logică, cât și cea fizică, sunt vizibile într-o singură figură.

Ca un exemplu, să considerăm cele patru rețele locale din fig. 4-49(a), în care opt dintre mașini aparțin VLAN-ului G (gri) și șapte aparțin VLAN-ului A (alb). Cele patru rețele locale sunt conectate cu două punți, B1 și B2. Dacă este folosită cablare centralizată cu fire torsadate, pot fi de asemenea prezente 4 hub-uri (care nu sunt prezentate în figură), dar la nivel logic un cablu cu mai mulți conectori și un hub sunt același lucru. Prezentarea lor în modul în care sunt figurați aici face figura mai puțin încărcată. De asemenea, termenul de punte tinde să fie folosit în zilele noastre mai ales în cazurile când există mai multe mașini pe fiecare port, ca în această figură, dar în rest termenii "punte" și "comutator" sunt interschimbabili. Fig. 4-49(b) prezintă aceleași mașini și aceleași același VLAN-uri folosind comutatoare cu un singur calculator pe fiecare port.

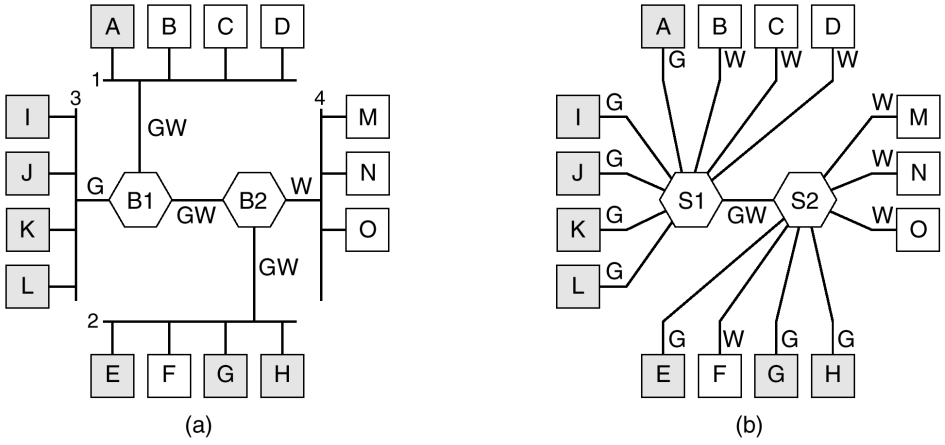


Fig. 4-49. (a) Patru rețele fizice organizate în două VLAN-uri, gri și alb, de către două punți (b) Aceleași 15 mașini organizate în doua VLAN-uri cu comutatoare

Pentru a asigura funcționarea corectă a VLAN-urilor, trebuie create tabele de configurare în comutatoare sau în punți. Aceste tabele stabilesc care VLAN este accesibil pe fiecare dintre porturi (linii). Atunci când un cadru este recepționat de la, să spunem, VLAN-ul gri, acesta trebuie înaintat către toate porturile marcate cu G. Acest lucru este valabil atât pentru traficul direcționat, cât și pentru cel cu destinație multiplă și cu difuzare.

Observați faptul că un port poate fi marcat cu mai multe culori de VLAN-uri. Acest lucru poate fi văzut clar în fig. 4-49(a). Să presupunem că mașina A difuzează un cadru. Puntea B1 recepționează cadrul și observă că acesta este provenit de la o stație din VLAN-ul gri, deci îl va înainta către toate porturile marcate cu G (cu excepția portului de unde a venit). Din moment ce B1 are numai două alte porturi și ambele sunt marcate cu G, cadrul va fi trimis pe ambele porturi.

În cazul lui B2 povestea este diferită. Aici puntea știe că nu există mașini gri in rețeaua locală 4, deci cadrul nu va fi înaintat acolo. Acesta va merge numai către rețeaua locală 2. Dacă unul dintre utilizatorii din rețeaua locală 4 își va schimba departamentul și va fi mutat în VLAN-ul gri, atunci tabela din interiorul lui B2 va trebui actualizată pentru a reeticheta portul cu GA în loc de A. Dacă mașina F devine gri, atunci portul către rețeaua locală 2 trebuie etichetat cu G în loc de GA.

Acum să presupunem că toate mașinile atât din rețeaua locală 2 cât și din rețeaua locală 4 devin gri. Atunci nu numai că porturile lui B2 către rețelele 2 și 4 vor fi marcate cu G, dar și portul lui B1 către B2 trebuie de asemenea reetichetat de la GA la G, din moment ce cadrele albe care ajung la B1 din rețelele 1 și 3 nu mai sunt înaintate către B2. În fig. 4-49(b) acești situație rămâne în picioare, numai că aici toate porturile care ajung la câte o singură mașină sunt etichetate cu o singură culoare deoarece acolo există un singur VLAN.

Până acum am presupus ca punțile și comutatoarele știu cumva ce culoare are un cadru recepționat. Cum știu acest lucru? Există trei metode folosite:

1. Fiecărui port îi este asociată o culoare de VLAN
2. Fiecărei adrese MAC îi este asociată o culoare de VLAN
3. Fiecărui protocol de nivel 3 sau fiecărei adrese IP îi este asociată o culoare de VLAN

Cu prima metodă, fiecare port este etichetat cu o culoare de VLAN. Totuși, această metodă funcționează doar dacă toate mașinile de pe un port aparțin aceluiași VLAN. În fig. 4-49(a), acest lucru este valabil în cazul lui B1 pentru portul către rețeaua 3, dar nu și pentru portul către rețeaua 1.

În cazul celei de-a doua metode, puntea sau comutatorul are o singură tabelă ce conține adresa MAC pe 48 de biți a fiecărui mașini conectate la el, împreună cu VLAN-ul căruia îi aparține mașina respectivă. În aceste condiții, este posibilă combinarea mai multor VLAN-uri pe o singură rețea locală fizică, cum este cazul rețelei 1 din fig. 4-49(a). Când un cadru este recepționat, tot ce trebuie să facă puntea sau comutatorul este să extragă adresa MAC și să caute intrarea corespunzătoare din tabelă, pentru a găsi VLAN-ul de unde a fost recepționat cadrul.

Cea de-a treia metodă presupune ca puntea sau comutatorul să examineze câmpul încărcare utilă al cadrului cu scopul de a clasifica, de exemplu, toate mașinile IP ca aparținând unui VLAN și toate mașinile AppleTalk ca aparținând altuia. Pentru cel dintâi, adresa IP poate fi de asemenea utilizată pentru identificarea mașinii. Această strategie este foarte utilă atunci când mai oricare din mai multe mașini sau calculatoare portabile pot fi cuplate în mai multe stații de ancorare. Din moment ce fiecare stație de ancorare are propria adresă MAC, doar cunoașterea stației de ancorare folosite nu spune nimic despre VLAN-ul căruia îi aparține laptop-ul.

Singura problemă cu această abordare este că nu respectă una dintre regulile de bază în rețele ce calculatoare: independența nivelurilor. Nu este treaba nivelului legătură de date ce este în câmpul de

încărcare utilă al cadrului. Acest nivel nu ar trebui să examineze aceste câmp și cu atât mai puțin să ia decizii pe baza conținutului acestuia. O consecință a utilizării acestei abordări este aceea că o modificare a unui protocol de nivel 3 (de exemplu o trecere de la IPv4 la IPv6) va duce la nefuncționarea comutatorului. Din nefericire, există pe piață comutatoare care funcționează în acest fel.

Desigur, nu este nimic în neregulă în rutarea bazată pe adrese IP – aproape tot cap. 5 este dedicat rutării IP – dar să combini nivelurile înseamnă să o cauți cu lumânarea. Un producător de comutatoare poate desconsidera acest argument susținând că toate comutatoarele comercializate de el înțeleg atât IPv4, cât și IPv6, deci totul este în regulă. Dar ce se va întâmpla atunci când va apărea IPv7? Producătorul probabil că va răspunde: cumpărați comutatoare noi, este asta atât de rău?

Standardul IEEE 802.1Q

Dacă ne gândim mai bine, ceea ce contează cu adevărat este VLAN-ul cadrului însuși, nu VLAN-ul mașinii care l-a trimis. Dacă ar exista o modalitate de identificare a VLAN-ului în antetul cadrului, atunci necesitatea de a examina câmpul încărcare ar dispărea. Pentru un model nou de rețea locală, cum ar fi 802.11 sau 802.16, ar fi fost destul de ușor să fie adăugat numărul VLAN-ului în antet. De fapt, câmpul *identificator de conexiune* din 802.16 este oarecum similar cu spiritul identificatorilor de VLAN. Dar ce să facem cu Ethernetul, care este tehnologia dominantă de rețele locale și care nu are câmpuri goale disponibile care să poată fi utilizate pentru identificatorul de VLAN?

Comitetul IEEE 802 a confruntat această problemă în 1995. După multe discuții, a făcut inimaginabilul și a modificat cadrul Ethernet. Noul format a fost publicat în standardul IEEE 802.1Q, lansat în 1998. Noul format conține marcajul pentru VLAN; îl vom examina în curând. Nu în mod surprinzător, schimbarea a ceva atât de bine împământenit cum este Ethernetul nu este în întregime trivială. O serie de întrebări care ne vin în gând sunt:

1. Trebuie să aruncăm câteva sute de milioane de plăci de rețea Ethernet?
2. Dacă nu, cine generează noul câmp?
3. Ce se întâmplă cu cadrele care au deja lungimea maximă?

Desigur, comitetul 802 a fost conștient (în mod dureros) de aceste probleme și a trebuit să ofere soluții, ceea ce a și făcut.

Cheia pentru găsirea soluției este să realizăm că identificatorii de VLAN sunt utilizați efectiv numai de punți și de comutatoare și nu de către mașinile utilizatorilor. Prin urmare, în fig. 4-49 nu este esențial ca identificatorii să fie prezenți pe liniile ce pornesc de la stații, atât timp cât sunt prezenți pe liniile ce interconectează punțile. Prin urmare, pentru a folosi VLAN-uri, punțile și comutatoarele trebuie să fie conștiente de existența acestora, dar aceasta era deja o cerință. Acum introducem necesitatea suplimentară ca acestea să implementeze 802.1Q, iar cele noi deja fac acest lucru.

La întrebarea dacă trebuie aruncate toate plăcile Ethernet, răspunsul este nu. Aduceți-vă aminte: comisia 802.3 nu a putut convinge oamenii să schimbe câmpul tip intr-un alt câmp numit lungime. Va puteți imagina reacția acestora la anunțul ca toate plăcile Ethernet au fost scoase din uz. Oricum, în momentul în care noile placi Ethernet vor apărea pe piață, se spera că acestea vor suporta 802.1Q și se vor putea integra în totalitate în VLAN-uri.

Așa că, daca cel care generează mesajul nu introduce câmpurile pentru VLAN, atunci cine o va face? Răspunsul este că prima punte sau comutator ce suporta VLAN la care ajunge un cadru, adaugă câmpurile, și la ultimul le scoate. Dar cum știe care cadru aparține cărui VLAN ? Ei bine, prima punte sau primul comutator poate atribui un număr VLAN unui port, se poate uita la adresa MAC, sau să examineze informația utilă. Până când toate plăcile Ethernet vor în conformitate cu 802.1Q,

suntem oarecum tot în punctul din care am plecat. Marea speranță este că de la început toate plăcile gigabit Ethernet vor fi în conformitate cu 802.1Q și pe măsură ce oamenii vor trece la gigabit Ethernet , 802.1Q va fi introdus automat. În ce privește problema cadrelor mai mari de 1518 octeți, 802.1Q ridică limita la 1522 octeți.

În timpul procesului de tranziție, multe rețele vor avea ca mașini perimate (de obicei Ethernet clasic sau rapid) care nu suporta VLAN și mașini (de obicei gigabit Ethernet) care suporta. Situația este arătată în fig. 4-50, unde simbolurile umbrite suportă VLAN iar celelalte nu. Pentru a simplifica problema, presupunem că toate comutatoarele suporta VLAN. Chiar dacă nu este cazul, primul comutator ce suportă VLAN poate adăuga marcaje bazate pe adrese MAC sau IP.

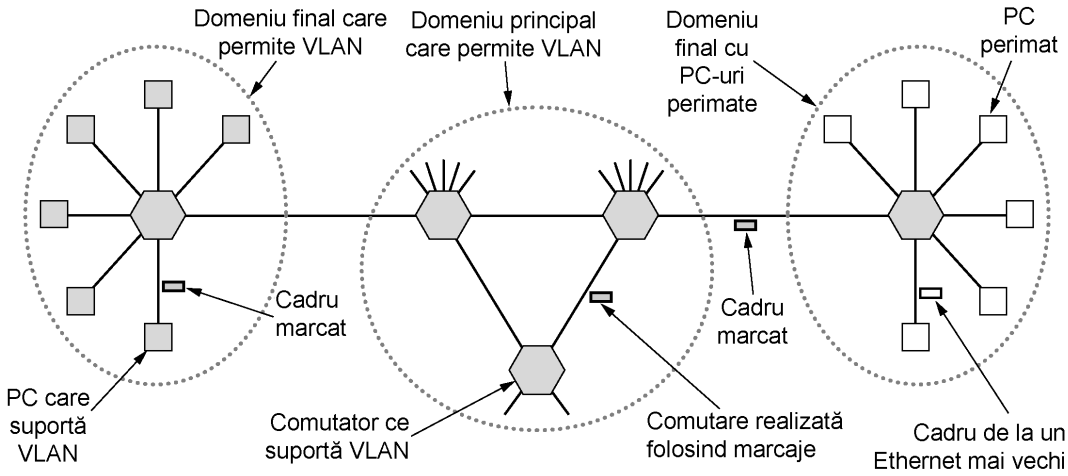

Domeniu final care permite VLAN
Domeniu principal care permite VLAN
Domeniu final cu PC-uri perimate
PC perimat
Cadru marcat
Cadru marcat
PC care suportă VLAN
Comutator ce suportă VLAN
Comutare realizată folosind marcaje
Cadru de la un Ethernet mai vechi

Fig. 4-50. Tranziția de la un Ethernet perimat la un Ethernet ce suportă VLAN. Simbolurile umbrite suportă VLAN; cele goale nu.

În aceasta figură, plăcile Ethernet ce suportă VLAN generează direct cadre marcate (de exemplu 802.1Q), și comutările ulterioare se folosesc de aceste marcaje. Pentru a face această comutare, comutatoarele trebuie sa știe în prealabil care VLAN poate fi accesat și pe ce port. Știind că un cadru aparține unui VLAN gri , nu ajută prea mult faptul ca un cadru aparține unui VLAN gri, până când comutatorul știe care porturi sunt conectate la mașinile din VLAN-ul gri. Așa că, comutatorul are nevoie de o tabela indexată de VLAN care să spună ce porturi sa folosească și care suporta VLAN sau nu.

Când un PC perimat trimite un cadru către un comutator ce suporta VLAN, comutatorul construiește un nou cadru marcat bazat pe cunoștințele sale despre VLAN-ul care l-a trimis (folosind portul, adresa MAC sau adresa IP). Din acel punct, nu mai contează dacă cel care trimite este o mașină legacy (perimată). Similar, un comutator care trebuie să trimită un cadru marcat către o mașină perimată (legacy) trebuie să reconstruiască cadrul în forma veche înainte de a-l furniza.

Haideți să privim formatul cadrului 802.1Q. Este schițat în fig. 4-51. Singura schimbare este adăugarea unei perechi de câmpuri a cate 2 octeți. Primul este identificatorul protocolului VLAN. El are întotdeauna valoarea 0x8100. Întrucât acest număr este mai mare de 1500, toate plăcile Ethernet interpretează acest număr ca tip nu ca lungime. Ce face o placa mai veche cu un asemenea cadru este o problemă deoarece asemenea cadre nu ar trebui trimise către acestea.

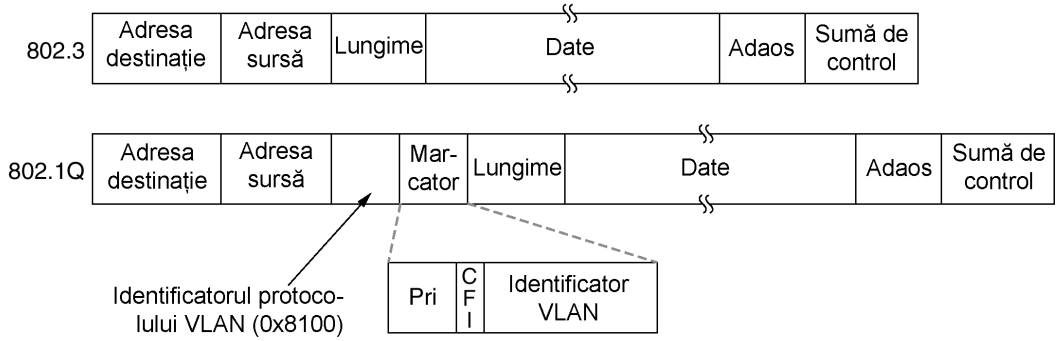


Fig. 4-51. Formatul cadrelor Ethernet 802.3 moștenite și 802.1Q.

Al doilea câmp de 2 octeți conține trei sub-câmpuri. Sub-câmpul principal este identificatorul VLAN ce ocupa cei mai puțin semnificativi 12 octeți. Aceasta este problema principala: cărui VLAN aparține fiecare cadru? Câmpul Prioritate de 3 biți nu are nici o legătură cu VLAN-ul, dar întrucât schimbarea antetului Ethernet este un eveniment foarte rar care s-ar desfășura pe parcursul a trei ani și ar implica 100 de oameni, de ce nu am pune alte informații folositoare în el? Acest câmp face posibila distingerea intre traficul în timp real implementat hard și cel implementat soft și de traficul intens pentru o mai buna calitate a serviciilor în Ethernet. Este nevoie de voce prin Ethernet (ca sa fim imparțiali, IP a avut un câmp similar mai mult de un sfert de secol și nu a fost folosit niciodată).

Ultimul bit, CFI (Canonical Format Indicator, rom: indicator de format canonic) ar fi trebuit sa fie numit CEI (Corporate Ego Indicator, rom: indicator de ego al corporației). Originar era folosit să indice adresele MAC în format little-indian sau big-indian însă această obișnuință s-a pierdut datorită controverselor. În zilele noastre prezența lui indică faptul că informațiile utile conțin un 802.5 cadru prestabilit (freezed-dried, rom: înghețat și uscat) care este transportat de o rețea Ethernet care speră să găsească la destinație un LAN 802.5. Tot acest aranjament, nu are nici o legătura cu VLAN-urile. Însă politica comitetului de standarde este foarte asemănătoare cu politica obișnuită: dacă votezi în favoarea bitului meu, votez și eu în favoarea bitului tău.

Așa cum am precizat mai sus, când un cadru marcat ajunge la un comutator ce suporta VLAN, comutatorul folosește, ca un index intr-o tabela identificatorul VLAN, pentru a găsi la ce port sa trimită. Dar de unde vine tabela? Este construita manual, ne-am întors de unde am plecat: configurarea manuala a punților. Frumusețea punților transparente este faptul că acestea sunt montate și pornite (plug-and-play) și nu au nevoie de configurare manuală. Ar fi păcat să se piardă această facilitate. Din fericire, punțile care suportă VLAN se pot autoconfigura pe baza marcajelor care vin. Dacă un cadru marcat, cum ar fi VLAN 4, ajunge la portul 3, aparent câteva calculatoare de pe portul 3 aparțin VLAN-ului 4. Standardul 802.1Q explica cum să construiești dinamic tabelele, în marea majoritate a cazurilor referindu-se la părți apropiate din algoritmul lui Perlman standardizat în 802.1D.

Înainte de a părăsi subiectul referitor la rutarea în VLAN, merita să facem o ultima observație. Mulți oameni în lumea Internetului și a Ethernetului susțin fanatic rețelele neorientate pe conexiune și se opun cu violență conexiunilor la nivelul legăturii de date. În momentul de față, VLAN-urile introduc ceva ce este surprinzător de asemănător cu o conexiune. Pentru a utiliza corect VLAN-uri, fiecare cadru are un nou identificator special care este utilizat pe post de index într-o tabelă din comutator, pentru a găsi destinația cadrului. Este exact același principiu de funcționare ce apare la rețelele orientate pe conexiune. În rețelele neorientate pe conexiune adresa destinație este folosită la rutare și nu exista identificatori de conexiune. Alte aspecte privind comunicarea vor fi tratate în cap. 5.

4.8 REZUMAT

Anumite rețele au un singur canal care este folosit pentru toate comunicațiile. În aceste rețele, problema principală de proiectare este alocarea acestui canal între stațiile concurente care doresc să îl folosească. Au fost puși la punct numeroși algoritmi de alocare a canalului. Un rezumat al unora dintre cele mai importante metode de alocare a canalului este prezentat în fig. 4-52.

Metodă	Descriere
FDM	Dedică o bandă de frecvență fiecărei stații
WDM	O schemă dinamică FDM pentru fibră optică
TDM	Dedică o cuantă de timp fiecărei stații
ALOHA pur	Transmisie nesincronizată în orice moment
ALOHA cuantificat	Transmisie aleatoare în cuante de timp bine definite
CSMA 1-persistent	Acces multiplu standard cu detectarea purtătoarei
CSMA nepersistent	Întârziere aleatoare când canalul este ocupat
CSMA p-persistent	CSMA cu probabilitatea de persistență p
CSMA/CD	CSMA cu oprire în cazul detectării unei coliziuni
Hartă de biți (bit map)	Utilizează o hartă de biți pentru planificare de tip rulare prin rotație
Numărare binară inversă	Următoarea este stația pregătită cu cel mai mare număr
Parcurgere arborescentă	Reduce conflictele prin activare selectivă
Divizarea lungimii de undă	Schemă FDM dinamică pentru fibre optice
MACA, MACAW	Protocoale LAN fără fir
Ethernet	CSMA/CD cu algoritm cu regresie exponențială binară
FHSS	Frequency hopping spread spectrum
DSSS	Direct sequence spread spectrum
CSMA/CA	Acces multiplu cu sesizarea purtătoarei cu evitarea coliziunilor

Fig. 4-52. Metode și sisteme de alocare a canalului pentru un canal obișnuit.

Cele mai simple scheme de alocare sunt FDM și TDM. Acestea sunt eficiente atunci când numărul de stații este mic și traficul continuu, oarecum echilibrat. Amândouă sunt larg folosite în aceste condiții, de exemplu pentru a diviza banda de legătură utilizată pentru trunchiuri telefonice.

Dacă numărul stațiilor este mare și variabil, iar traficul de tip rafală, atunci FDM și TDM nu sunt alegeri bune. Ca alternativă a fost propus protocolul ALOHA, cu sau fără cuantificare și control. ALOHA și numeroasele sale variante și derivate a fost pe larg discutat, analizat și folosit în sisteme reale.

Atunci când starea canalului poate fi detectată, stațiile pot evita începerea unei transmisii cât timp transmite altă stație. Această tehnică, detectarea purtătoarei, a condus la o diversitate de protocoale care pot fi folosite pe LAN-uri și MAN-uri.

Există o clasă de protocoale care elimină total conflictele, sau cel puțin le reduce considerabil. Numărarea binară inversă elimină complet conflictele. Protocolul de parcurgere arborescentă le reduce împărțind dinamic stațiile în două grupuri disjuncte, unuia permițându-i-se să transmită iar celuilalt nu. Acesta încearcă să facă împărțirea astfel, încât transmisia să-i fie permisă unei singure stații dintre cele pregătite să trimită.

LAN-urile fără fir au propriile lor probleme și soluții. Cea mai mare problemă este cauzată de stații ascunse, astfel încât CSMA nu funcționează. O clasă de soluții, tipizată de MACA și MACAW, intenționează să stimuleze transmisiile în jurul destinației, pentru a îmbunătăți funcționarea CSMA.

FHSS și DSSS sunt de asemenea utilizate. IEEE 802.11 combină CSMA și MACAW pentru a produce CSMA/CA.

Ethernetul este forma dominanta pentru rețele locale. Acesta utilizează CSMA/CD pentru alocarea canalului. Versiunile vechi foloseau cabluri ce șerpuiau de la o mașina la alta, dar acum sunt utilizate perechi de fire torsadate ce se conectează în noduri și comutatoare. Vitezele au crescut de la 10 Mbps la 1 Gbps și cresc în continuare.

LAN-urile fără fir sunt din ce în ce mai comune, 802.11 dominând acest domeniu. Nivelul său fizic permite cinci moduri de transmisie diferite, incluzând infraroșu, o varietate de spread spectrum schemes, și un sistem FDM multicanal. Poate opera cu câte o stație de bază în fiecare celulă, dar poate opera și fără nici. Protocolul este o versiune de MACAW cu sesizarea virtuală a purtătoarei. MAN-urile fără fir au început deja să apară. Acestea sunt sisteme de bandă largă care utilizează unde radio pentru a înlocui ultimele porțiuni în conexiunile telefonice. Sunt folosite tehnici tradiționale de modulație de bandă îngustă. Calitatea serviciilor este importantă, cu 802.16 definindu-se patru clase și anume: viteză de transmisie constantă, două viteze de transmisie variabile și o viteză de transmisie cu cea mai bună încercare (eng. best efforts).

Sistemul Bluetooth este de asemenea fără fir dar este adresat mai mult către sistemele desktop, pentru conectarea căștilor și a altor echipamente la calculatoare fără a utiliza fire. Se intenționează de asemenea conectarea perifericelor, cum ar fi faxurile la telefoane mobile. Ca și 802.11, acesta folosește FHSS în banda ISM. Datorită nivelului de zgomot din multe medii și datorită necesității unei interacțiuni în timp real, diferite protocoale înglobează mecanisme complicate pentru urmărirea și corecția erorilor.

Având atât de multe LAN-uri diferite, este necesară o metodă de a le interconecta. Punțile și comutatoarele sunt folosite în acest scop. Algoritmul cu arbore de acoperire este folosit pentru a construi punți plug-and-play. O noua dezvoltare în domeniul interconectării LAN-urilor este VLAN, care separă topologia logică a LAN-urilor de topologia fizică. Un nou format pentru cadrele Ethernet (802.1Q) a fost introdus pentru a oferi o modalitate mai simplă de introducere a VLAN-urilor în organizații.

4.9 PROBLEME

1. Pentru această problema folosiți o formula din acest capitol, însă înainte de a începe rezolvarea problemei scrieți formula. Cadrele ajung aleator la un canal de 100 Mbps pentru transmitere. Dacă în momentul când un cadru ajunge avem canalul ocupat, acesta își așteaptă rândul într-o coadă. Dimensiunea cadrului este distribuita exponential cu o medie de 10.000 biți/cadru. Pentru fiecare din următoarele rate de sosire, precizați întârzierea medie a unui cadru, incluzând timpul cât acesta stă în coadă și timpul cât durează transmisia.

a) 90 cadre/sec.

b) 900 cadre/sec.

c) 9000 cadre/sec.

2. Un grup de N stații folosesc în comun un canal ALOHA pur de 56 Kbps. Fiecare stație emite în medie un cadru de 1000 de biți la fiecare 100 sec, chiar dacă cel precedent nu a fost încă trimis (de exemplu, stațiile folosesc zone tampon). Care este valoarea maximă a lui N?

3. Comparați întârzierea unui canal ALOHA pur cu aceea a unui canal ALOHA cuantificat la încărcare mică. Care dintre ele este mai mică? Motivați răspunsul.

4. Zece mii de stații de rezervare a biletelor de avion concurează pentru folosirea unui singur canal ALOHA cuantificat. O stație obișnuită face 18 cereri/oră. O cuantă este de 125 μs. Care este încărcarea totală aproximativă a canalului?

5. O populație mare de utilizatori ALOHA generează 50 cereri/sec, inclusiv originalele și retransmisiile. Timpul este cuantificat în unități de 40 ms.
a) Care este șansa de succes a primei încercări?
b) Care este probabilitatea unui număr de exact k coliziuni urmate de un succes?
c) Câte încercări de transmisie ne așteptăm să fie necesare?

6. Măsurătorile făcute asupra unui canal ALOHA cuantificat, cu un număr infinit de utilizatori, arată că 10% din cuante sunt nefolosite.
a) Care este încărcarea canalului, G?
b) Care este productivitatea?
c) Canalul este subîncărcat sau supraîncărcat?

7. Într-un sistem cuantificat ALOHA cu o populație infinită, numărul mediu de cuante pe care o statie le așteaptă între o coliziune și retransmisia ei, este 4. Reprezentați curba întârzierii în funcție de productivitate, pentru acest sistem.

8. Cat timp o stație s trebuie să aștepte în cel mai rău caz înainte de a putea transmite cadre intr-un LAN ce folosește:
a) protocolul de bază harta de biți?
b) protocolul lui Mok și Ward cu permutare virtuală a numerelor stațiilor?

9. Un LAN folosește versiunea lui Mok și Ward pentru numărătoarea inversă binară. La un anumit moment, cele zece stații au numerele virtuale de stație 8, 2, 4, 5, 1, 7, 3, 6, 9 și 0. Următoarele trei stații care trebuie să emită sunt 4, 3 și 9, în această ordine. Care sunt noile numere virtuale de stație după ce toate cele trei și-au terminat transmisiile?

10. Șaisprezece stații concurează pentru folosirea unui canal comun folosind protocolul cu parcurgere arborescentă adaptivă. Dacă toate stațiile ale căror adrese sunt numere prime devin brusc simultan disponibile, câte intervale de bit sunt necesare pentru a rezolva conflictul?

11. O colecție de 2^n stații folosesc protocolul cu parcurgere arborescentă adaptivă pentru a arbitra accesul la un cablu comun. La un moment dat, două dintre ele devin disponibile. Care este numărul minim, maxim și mediu de cuante pentru a parcurge arborele dacă $2^n >> 1$?

12. LAN-urile fără fir pe care le-am studiat foloseau protocoale ca MACA în loc de CSMA/CD. În ce condiții ar fi posibil să folosească CSMA/CD?

13. Care sunt caracteristicile comune ale protocoalelor de acces la canal WDMA și GSM?

14. Șase stații , de la A la F, comunica utilizând protocolul MACA. Este posibil ca doua transmisii sa aibă loc simultan? Explicați răspunsul.

15. O clădire cu 7 etaje are 15 birouri alăturate pe fiecare etaj. Fiecare birou conține o priză de perete pentru un terminal pe peretele din față, astfel încât prizele formează o rețea rectangulară în plan vertical, cu o distanță de 4 m între prize, atât pe orizontală cât și pe verticală. Presupunând că este posibil să se monteze câte un cablu direct între orice pereche de prize, pe orizontală, verticală sau diagonală, câti metri de cablu sunt necesari pentru conectarea tuturor prizelor folosind:
a) O configurație stea cu un singur ruter în mijloc?
b) Un LAN 802.3?
c) O rețea de tip inel (fără fir central)?

16. Care este viteza (în bauds) a unui LAN 802.3 standard de 10 Mbps?

17. Schițați codificarea Manchester pentru șirul de biți: 0001110101.

18. Schițați codificarea Manchester diferențială pentru șirul de biți din problema precedentă. Presupuneți că linia este inițial în stare jos.

19. Un LAN CSMA/CD de 10 Mbps (care nu e 802.3), lung de 1 km, are o viteză de propagare de 200 m/μs. Cadrele de date au o lungime de 256 biți, incluzând 32 de biți de antet, suma de control și alte date suplimentare. Primul interval de bit după o transmitere efectuată cu succes este rezervat pentru receptor spre a ocupa canalul pentru a trimite un cadru de confirmare de 32 de biți. Care este viteza efectivă de date, excluzând încărcarea suplimentară și presupunând că nu sunt coliziuni?

20. Două stații CSMA/CD încearcă să transmită fiecare fișiere mari (multicadru). După ce este trimis fiecare cadru, ele concurează pentru canal folosind algoritmul de regresie exponențială binară. Care este probabilitatea terminării conflictului la runda k, și care este numărul mediu de runde per conflict?

21. Să considerăm cazul unei rețele CSMA/CD de 1G bps, cu un cablu mai lung de 1 km, fără repetoare. Viteza semnalului pe cablu este de 200.000 km/s. Care este dimensiunea minimă a cadrului?

22. Un pachet IP ce trebuie transmis în Internet are 60 octeți cu tot cu antete. Daca LLC nu este utilizat, este nevoie să se adauge informație de umplutură în cadrul Ethernet, și dacă da, câți octeți?

23. Cadrele Ethernet trebuie să aibă o lungime minimă de 64 de octeți pentru a avea siguranța că emițătorul încă mai emite, în cazul unei coliziuni la capatul celălalt al cablului. Fast Ethernet-ul are aceeași dimensiune minimă a cadrului de 64 de octeți, dar poate emite biții de zece ori mai rapid. Cum este posibil să se mențină aceeași dimensiune minimă a cadrului?

24. Autorii unor cărți susțin că dimensiunea maximă a cadrului Ethernet este de 1518 octeți în loc de 1500 octeți. Au aceștia dreptate? Explicați răspunsul.

25. Specificațiile 1000Base-SX spun că ceasul ar trebui să meargă la 1250 MHz, deși Gigabit Ethernet ar trebui sa transmită 1 Gbps. Este folosit acest plus de viteză pentru a mări siguranța transmisiei? Dacă nu, specificați ce se întâmplă.

26. Câte cadre pe secundă poate manevra gigabit Ethernet? Luați în considerare toate cazurile relevante. Sugestie: contează faptul că este o rețea gigabit Ethernet.

27. Numiți două rețele care permit sa aibă cadre împachetate cap-la-cap. De ce se merita să ai această facilitate?

28. În fig. 4-27 sunt arătate patru stații, A, B, C și D. Care dintre ultimele doua stații credeți ca este mai aproape de A și de ce?

29. Presupunând ca un 11-Mbps LAN 802.11b transmite cadre de 64-octeti cap-la-cap printr-un canal radio rata erorilor de 10^{-7} . Câte cadre pe secundă vor fi distruse în medie?

30. O rețea 802.16 are lungimea canalului de 20 MHz. Câți biți/sec pot fi transmiși la o stație conectată?

31. IEEE 802.16 suportă patru clase de servicii. Care clasă este cea mai bună alegere pentru a transmite semnal video necomprimat?

32. Dați două motive pentru care rețelele ar trebui să utilizeze corectarea erorilor în loc de detecția erorilor și retransmisia datelor?

33. În fig. 4-35, am văzut ca un dispozitiv Bluetooth poate fi în două piconet-uri în același timp. Există vreun motiv ca un dispozitiv sa nu fie stăpân în ambele piconet-uri în același timp?

34. Fig. 4-25 arată diferite protocoale de nivel fizic. Care dintre acestea este mai apropiat de protocolul de nivel fizic al Bluetooth? Care este marea diferență dintre cele doua?

35. Bluetooth suportă două tipuri de legătură între un stăpân și un sclav. Care sunt acestea și la ce sunt folosite fiecare?

36. Cadrul de semnalizare la FHSS (frequency hopping spread spectrum) varianta 802.11 conține timpul de locuire (dwell time). Credeți ca la Bluetooth, cadrul de semnalizare analog, conține de asemenea timpul de locuire (dwell time)? Discutați răspunsul.

37. Considerați LAN-urile interconectate din fig. 4-44. Presupuneți că gazda a și b sunt în LAN-ul 1, c este în LAN-ul 2 și d este în LAN-ul 8. Inițial tabelele de dispersie din toate punțile sunt goale și se folosește arborele de acoperire din fig. 4-44(b). Arătați cum tabelele de dispersie din punți diferite se schimbă după fiecare din următoarele evenimente ce se succed : primul a, apoi b și așa mai departe.
a) a trimite către d.
b) c trimite către a.
c) d trimite către c.
d) d trimite către LAN-ul 6.
e) d trimite către a.

38. O consecință în folosirea unui arbore de acoperire pentru a retransmite cadre intr-un LAN extins este ca unele punți nu participă la retransmiterea cadrelor. Identificați trei punți de acest fel în fig. 4-44. Exista vreun motiv pentru a păstra aceste punți, chiar daca ele nu sunt folosite pentru retransmitere?

39. Imaginați-vă ca un comutator are plăci de extensie pentru patru linii de intrare. Se întâmplă frecvent ca un cadru care ajunge pe una din aceste linii trebuie să iasă pe altă linie pe aceeași placă. Ce variante are proiectantul comutatorului pentru aceasta situație?

40. Un comutator proiectat pentru a fi utilizat cu un Ethernet rapid are un fund de sertar care poate transfera 10 Gbps. Câte cadre/sec poate manevra în cel mai rău caz?

41. Considerați rețeaua din fig. 4-49(a). Daca mașina J devine brusc albă; este nevoie de vreo schimbare la etichetare? Dacă da, ce anume?

42. Descrieți pe scurt diferențele dintre comutatoarele cu memorare și retransmitere și cele cu cut-through?

43. În ceea ce privește cadrele defecte, comutatoarele cu memorare și retransmitere au un avantaj fata de cele cut-through. Explicați care sunt acestea.

44. Pentru a pune în funcțiune VLAN-uri, este nevoie de tabele de configurație în comutatoare și punți. Ce s-ar întâmpla daca VLAN-urile din fig. 4-49(a) ar utiliza noduri în loc de mediu partajat? Nodurile au nevoie de tabele de configurare? De ce sau de ce nu?

45. În fig. 4-50 comutatorul din domeniul final cu PC îmbătrânite, figurat în dreapta este un comutator pregătit pentru VLAN. Este posibilă utilizarea unui comutator vechi în acest caz? Dacă da, cum va funcționa acesta? Dacă nu, de ce?

46. Scrieți un program care să simuleze comportamentul protocolului CSMA/CD în Ethernet când exista N stații pregătite sa transmită în timp ce se transmite un cadru. Programul vostru trebuie să prezinte timpii când fiecare stație începe sa transmită cu succes cadrul. Presupuneți că un tact de ceas apare odată la fiecare cuantă de timp (51,2 microsecunde) și o detecție de coliziune și o secvență de bruiaj durează o cuantă de timp. Toate cadrele sunt de dimensiune maximă admisă.

5

NIVELUL REȚEA

Nivelul rețea are ca sarcină preluarea pachetelor de la sursă și transferul lor către destinație. Ajungerea la destinație poate necesita mai multe salturi prin rutere intermediare de-a lungul drumului. Această funcție contrastează clar cu cea a nivelului legătură de date, care avea scopul mult mai modest de a transfera cadre de la un capăt al unui fir la celălalt. Astfel nivelul rețea este cel mai scăzut nivel care se ocupă de transmisii capăt la capăt.

Pentru realizarea scopurilor propuse, nivelul rețea trebuie să cunoască topologia subrețelei de comunicație (de exemplu mulțimea tuturor ruterelor) și să aleagă calea cea mai potrivită prin aceasta. De asemenea trebuie să aleagă căile de urmat astfel, încât să nu încarce excesiv unele legături de comunicație sau rutere în timp ce altele sunt inactive. În fine, când sursa și destinația fac parte din rețele diferite, apar probleme noi. Este sarcina nivelului rețea să se ocupe de ele. În acest capitol vom studia toate aceste aspecte și le vom exemplifica, in primul rând folosind Internetul și protocolul lui la nivelul rețea, IP, cu toate că vom vorbi și despre rețele fără fir.

5.1 CERINȚELE DE PROIECTARE ALE NIVELULUI REȚEA

Vom prezenta, în continuare, o introducere a cerințelor pe care proiectantul nivelului rețea trebuie să le rezolve. Acestea includ serviciile furnizate nivelului transport și proiectarea internă a subrețelei.

5.1.1 Comutare de pachete de tip Memorează-și-Retransmite (Store-and-Forward)

Dar înainte de a începe explicarea detaliilor nivelului rețea, merită probabil sa reinițializăm contextul în care operează protocoalele de la nivelul rețea. Acest context este prezentat în fig. 5-1. Componentele majore ale sistemului sunt echipamentul companiei de telecomunicații (rutere conectate prin linii de transmisie), prezentat în interiorul ovalului umbrit, și echipamentul clientului, prezentat în afara ovalului. Gazda *H1* este conectată direct la unul dintre ruterele companiei de telecomunicații, *A*, printr-o linie închiriată. În contrast, *H2* este într-o rețea LAN cu un ruter, *F*, deținut și operat de către client. Acest ruter are, deasemeni, și o linie închiriată către echipamentul companiei de telecomunicații. Am prezentat *F* ca fiind în afara ovalului, deoarece nu aparține companiei de telecomunicații, dar în termeni de construcție, software și protocoale, probabil că nu diferă față de ruterele acesteia. Este discutabil dacă aparține subrețelei, dar în contextul acestui capitol ruterele din localul clientului sunt considerate parte a subrețelei deoarece rulează aceeași algoritmi ca și ruterele companiei de telecomunicații (și aici principala noastră preocupare sunt algoritmii).

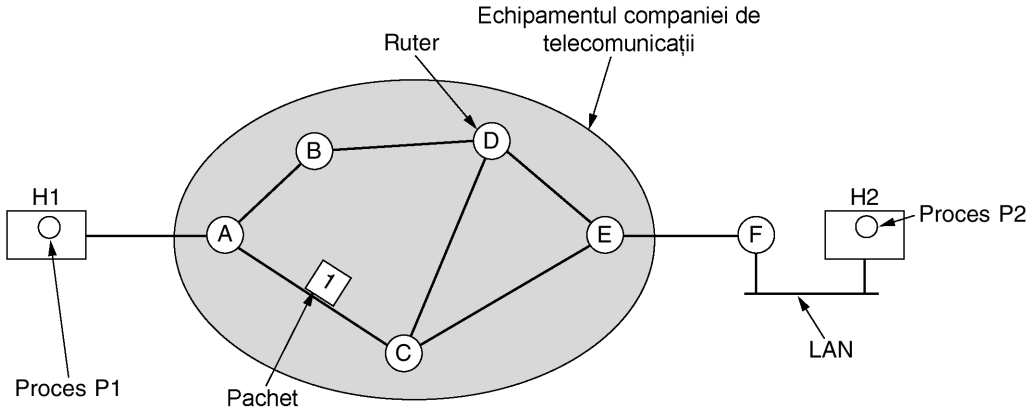


Fig. 5-1. Cadrul protocoalelor nivelului rețea.

Acest echipament este folosit după cum urmează. O gazdă care are de transmis un pachet îl transmite celui mai apropiat ruter, fie in aceeași rețea LAN, fie printr-o legătură punct la punct cu compania de telecomunicații. Pachetul este memorat acolo până ajunge integral, astfel încât să poată fi verificată suma de control. Apoi este trimis mai departe către următorul ruter de pe traseu, până ajunge la gazda destinație, unde este livrat. Acest mecanism reprezintă comutarea de pachete de tip memorează-și-retransmite, așa cum am văzut în capitolele anterioare.

5.1.2 Servicii furnizate nivelului transport

Nivelul rețea furnizează servicii nivelului transport la interfața dintre cele două niveluri. O întrebare importantă este ce fel de servicii furnizează nivelul rețea nivelului transport. Serviciile nivelului rețea au fost proiectate având în vedere următoarele scopuri:

1. Serviciile trebuie să fie independente de tehnologia ruterului.
2. Nivelul transport trebuie să fie independent de numărul, tipul și topologia ruterelor existente.

3. Adresele de rețea disponibile la nivelul transport trebuie să folosească o schemă de numerotare uniformă, chiar în cadrul rețelelor LAN și WAN.

Obiectivele fiind stabilite, proiectantul nivelului rețea are o mare libertate în a scrie specificațiile detaliate ale serviciilor oferite nivelului transport. Această libertate degenerează adesea într-o aprigă bătălie între două tabere opuse. Problema centrală a discuției este dacă nivelul rețea trebuie să furnizeze servicii orientate pe conexiune sau servicii neorientate pe conexiune.

O tabără (reprezentată de comunitatea Internet) afirmă că scopul ruterului este de a transfera pachete și nimic mai mult. În viziunea lor (bazată pe experiența a aproape 30 de ani de exploatare a unei rețele de calculatoare în funcțiune), subrețeaua este inerent nesigură, indiferent cum ar fi proiectată. De aceea calculatoarele gazdă trebuie să accepte faptul că rețeaua este nesigură și să facă controlul erorilor (i.e., detecția și corecția erorii) și controlul fluxului ele însele.

Acest punct de vedere duce rapid la concluzia că serviciul rețea trebuie să fie neorientat pe conexiune, cu două primitive SEND PACKET și RECEIVE PACKET și cu foarte puțin în plus. În particular, nu trebuie făcută nici o operație pentru controlul ordinii sau fluxului pachetelor pentru că oricum calculatorul gazdă va face acest lucru, și, de obicei, dublarea acestor operații aduce un câștig nesemnificativ. În continuare, fiecare pachet va trebui să poarte întreaga adresă de destinație, pentru că fiecare pachet este independent de pachetele predecesoare, dacă acestea există.

Cealaltă tabără (reprezentată de companiile de telefoane) afirmă că subrețeaua trebuie să asigure un serviciu orientat pe conexiune sigur. Ei susțin că 100 de ani de experiență cu sistemul telefonic mondial reprezintă un ghid excelent. În această perspectivă, calitatea serviciului este elementul dominant, și într-o subrețea fără conexiuni, calitatea serviciului este dificil de obținut, în special pentru trafic in timp real cum ar fi voce și imagine.

Aceste două tabere sunt cel mai bine exemplificate de Internet și rețele ATM. Rețeaua Internet oferă un serviciu la nivelul rețea neorientat pe conexiune; rețelele ATM oferă un serviciu la nivelul rețea orientat pe conexiune. Totuși, este interesant de notat că cu cât garantarea calității serviciului devine din ce in ce mai importantă, Internetul evoluează. În particular, începe să dobândească proprietăți asociate normal cu serviciile orientate conexiune, așa cum vom vedea mai târziu. De fapt, ne-am făcut o părere despre această evoluție in timpul studiului despre rețele VLAN în Cap. 4.

5.1.3 Implementarea serviciului neorientat pe conexiune

După ce am văzut cele două clase de servicii pe care nivelul rețea le furnizează utilizatorilor săi, este momentul să vedem funcționarea internă a acestui nivel. Sunt posibile două organizări diferite, în funcție de tipul serviciului oferit. Dacă este oferit un serviciu neorientat pe conexiune, atunci pachetele sunt trimise in subrețea individual și dirijate independent de celelalte. Nu este necesară nici o inițializare prealabilă. În acest context, pachetele sunt numite frecvent **datagrame** (datagrams) (prin analogie cu telegramele), iar subrețeaua este numită **subrețea datagramă** (datagram subnet). Dacă este folosit serviciul orientat conexiune, atunci, înainte de a trimite pachete de date, trebuie stabilită o cale de la ruterul sursă la ruterul destinație. Această conexiune este numită **VC** (**virtual circuit**, circuit virtual), prin analogie cu circuitele fizice care se stabilesc în sistemul telefonic, iar subrețeaua este numită **subrețea cu circuite virtuale** (**virtual-circuit subnet**). În această secțiune vom studia subrețele datagramă; în următoarea secțiune vom studia subrețelele cu circuite virtuale.

Să vedem cum funcționează o subrețea datagramă. Să presupunem că procesul *P1* din fig. 5-2 are un mesaj lung pentru procesul *P2*. El transmite mesajul nivelului transport, cu instrucțiunile de livrare către procesul *P2* aflat pe calculatorul gazdă *H2*. Codul nivelului transport rulează pe calculatorul

gazdă *H1*, de obicei în cadrul sistemului de operare. Acesta inserează la începutul mesajului un antet corespunzător nivelului transport și transferă rezultatul nivelului rețea, probabil o altă procedură din cadrul sistemului de operare.

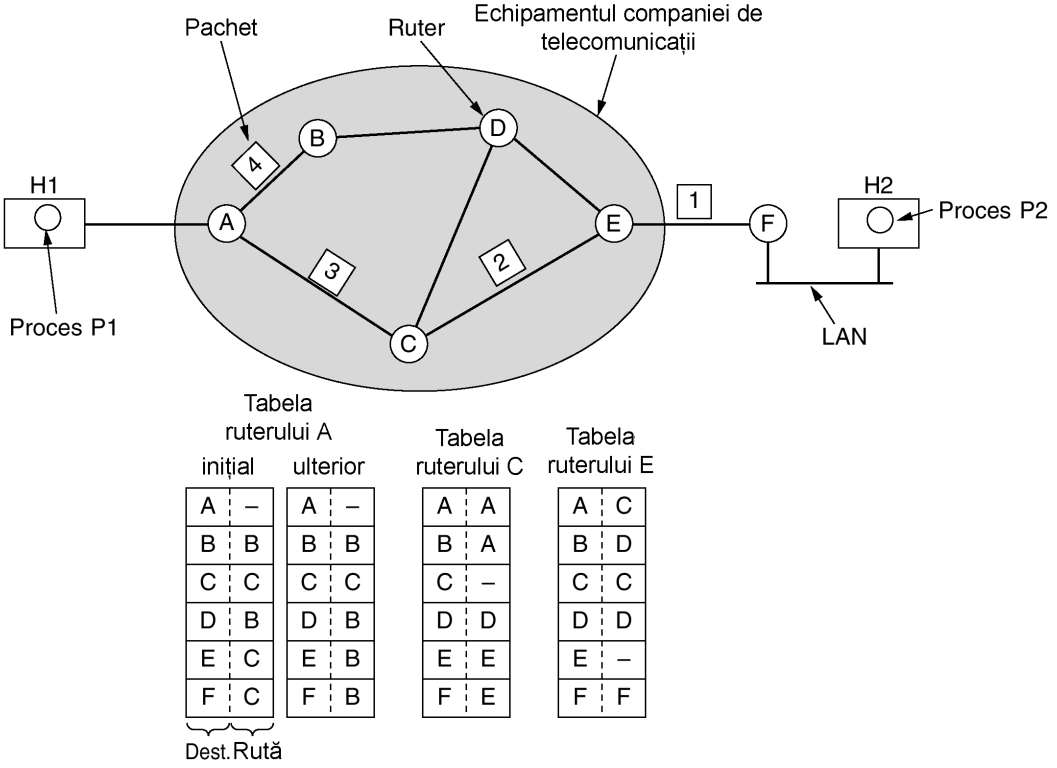


Fig. 5-2. Dirijarea într-o subrețea datagramă.

Să presupunem că mesajul este de patru ori mai lung decât dimensiunea maximă a unui pachet, așa că nivelul rețea trebuie să îl spargă în patru pachete, 1, 2, 3, și 4 și să le trimită pe fiecare în parte ruterului *A*, folosind un protocol punct-la-punct, de exemplu, PPP. Din acest punct controlul este preluat de compania de telecomunicații. Fiecare ruter are o tabelă internă care îi spune unde să trimită pachete pentru fiecare destinație posibilă. Fiecare intrare în tabelă este o pereche compusă din destinație și linia de ieșire folosită pentru acea destinație. Pot fi folosite doar linii conectate direct. De exemplu, în fig. 5-2, *A* are doar două linii de ieșire – către *B* și *C* – astfel că fiecare pachet ce vine trebuie trimis către unul dintre aceste rutere, chiar dacă ultima destinație este alt ruter. Tabela de rutare inițială a lui *A* este prezentată în figură sub eticheta „inițial”.

Cum au ajuns la *A*, pachetele 1, 2 și 3 au fost memorate pentru scurt timp (pentru verificarea sumei de control). Apoi fiecare a fost trimis mai departe către *C* conform tabelei lui *A*. Pachetul 1 a fost apoi trimis mai departe către *E* și apoi către *F*. Când a ajuns la *F*, a fost încapsulat într-un cadru al nivelului legătură de date și trimis către calculatorul gazdă *H2* prin rețeaua LAN.

Totuși, ceva diferit s-a întâmplat cu pachetul 4. Când a ajuns la *A* a fost trimis către ruterul *B,* chiar dacă și el este destinat tot lui *F*. Dintr-un motiv oarecare, *A* a decis să trimită pachetul 4 pe o rută diferită de cea urmată de primele trei. Poate că a aflat despre o congestie undeva pe calea *ACE*

și și-a actualizat tabela de rutare, așa cum apare sub eticheta „mai târziu”. Algoritmul ce administrează tabelele și ia deciziile de rutare se numește **algoritm de rutare** (routing algorithm). Algoritmii de rutare sunt unele dintre principalele elemente pe care le vom studia în acest capitol.

5.1.4 Implementarea serviciilor orientate pe conexiune

Pentru serviciile orientate conexiune, avem nevoie de o subrețea cu circuite virtuale. Să vedem cum funcționează aceasta. Ideea care se stă la baza circuitelor virtuale este evitarea alegerii unei noi căi (rute) pentru fiecare pachet trimis, ca în fig. 5-2. În schimb, atunci când se stabilește o conexiune, se alege o cale între mașina sursă și mașina destinație, ca parte componentă a inițializării conexiunii și aceasta este memorată în tabelele ruterelor. Acea cale este folosită pentru tot traficul de pe conexiune, exact în același mod în care funcționează sistemul telefonic. Atunci când conexiunea este eliberată, este închis și circuitul virtual. În cazul serviciilor orientate conexiune, fiecare pachet poartă un identificator care spune cărui circuit virtual îi aparține.

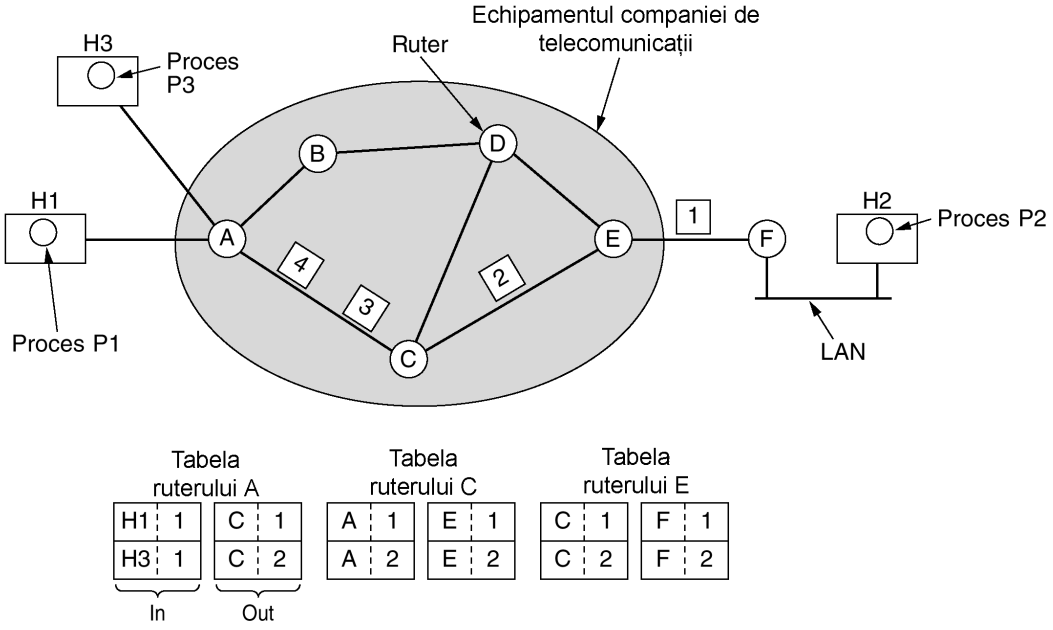


Fig. 5-3. Dirijare în cadrul unei subrețele cu circuite virtuale.

De exemplu, să considerăm situația din fig. 5-3. Aici calculatorul gazdă *H1* a stabilit conexiunea 1 cu calculatorul gazdă *H2*. Aceasta este memorată ca prima intrare în fiecare tabelă de rutare. Prima linie a tabelei lui *A* spune că dacă un pachet purtând identificatorul de conexiune 1 vine de la *H1*, atunci trebuie trimis către ruterul *C*, dându-i-se identificatorul de conexiune 1. Similar, prima intrare a lui C dirijează pachetul către *E*, tot cu identificatorul de conexiune 1.

Acum să vedem ce se întâmplă dacă *H3* vrea, de asemenea, să stabilească o conexiune cu *H2*. Alege identificatorul de conexiune 1 (deoarece inițializează conexiunea și aceasta este singura conexiune) și indică subrețelei să stabilească circuitul virtual. Aceasta conduce la a doua linie din tabele. Observați că apare un conflict deoarece deși *A* poate distinge ușor pachetele conexiunii 1 de la *H1*

de pachetele conexiunii 1 de la *H3*, *C* nu poate face asta. Din acest motiv, *A* asociază un identificator de conexiune diferit pentru traficul de ieșire al celei de a doua conexiuni. Pentru evitarea conflictelor de acest gen ruterele trebuie să poată înlocui identificatorii de conexiune în pachetele care pleacă. În unele contexte, aceasta se numește comutarea etichetelor (label switching).

5.1.5 Comparație între subrețele cu circuite virtuale și subrețele datagramă

Atât circuitele virtuale cât și datagramele au suporteri și oponenți. Vom încerca acum să rezumăm argumentele ambelor tabere. Principalele aspecte sunt prezentate în fig. 5-4, deși cei extrem de riguroși ar putea probabil găsi un contraexemplu pentru toate cele descrise în această figură.

Problemă	Subrețea datagramă	Subrețea cu circuite virtuale (CV)
Stabilirea circuitului	Nu este necesară	Obligatorie
Adresare	Fiecare pachet conține adresa completă pentru sursă și destinație	Fiecare pachet conține un număr mic de CV
Informații de stare	Ruterele nu păstrează informații despre conexiuni	Fiecare CV necesită spațiu pentru tabela ruterului per conexiune
Dirijare	Fiecare pachet este dirijat independent	Calea este stabilită la inițierea CV; toate pachetele o urmează
Efectul defectării ruterului	Nici unul, cu excepția pachetelor pierdute în timpul defectării	Toate circuitele virtuale care trec prin ruterul defect sunt terminate
Calitatea serviciului	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse
Controlul congestiei	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse

Fig. 5-4. Comparație între subrețele datagramă și subrețele cu circuite virtuale.

În interiorul subrețelei există situații în care trebuie să se aleagă între facilități antagoniste specifice fie circuitelor virtuale, fie datagramelor. Un astfel de compromis este acela între spațiul de memorie al ruterului și lățimea de bandă. Circuitele virtuale permit pachetelor să conțină numere de circuite în locul unor adrese complete. Dacă pachetul tinde să fie foarte mic, atunci existența unei adrese complete în fiecare pachet poate reprezenta o supraîncărcare (overhead) importantă și deci o irosire a lățimii de bandă. Prețul plătit pentru folosirea internă a circuitelor virtuale este spațiul necesar păstrării tabelei în ruter. Soluția mai ieftină este determinată de raportul între costul circuitelor de comunicație și cel al memoriei ruterului.

Alt compromis este cel între timpul necesar stabilirii circuitului și timpul de analiză a adresei. Folosirea circuitelor virtuale presupune existența unei faze inițiale de stabilire a căii, care cere timp și consumă resurse. Oricum, este ușor să ne imaginăm ce se întâmplă cu un pachet de date într-o subrețea bazată pe circuite virtuale: ruterul folosește numărul circuitului ca un index într-o tabelă pentru a afla unde merge pachetul. Într-o rețea bazată pe datagrame, pentru a găsi intrarea corespunzătoare destinației se folosește o procedură de căutare mult mai complicată.

O altă problemă este cea a dimensiunii spațiului necesar pentru tabela din memoria ruterului. O subrețea datagramă necesită o intrare pentru fiecare destinație posibilă, în timp ce o rețea cu circuite virtuale necesită o intrare pentru fiecare circuit virtual. Totuși, acest avantaj este relativ iluzoriu deoarece și pachetele de inițializare a conexiunii trebuie rutate, iar ele folosesc adresele destinație, la fel ca și datagramele.

Circuitele virtuale au unele avantaje în garantarea calității serviciului și evitarea congestionării subrețelei, deoarece resursele (de exemplu zone tampon, lărgime de bandă și cicluri CPU) pot fi rezervate în avans, atunci când se stabilește conexiunea. La sosirea pachetelor, lățimea de bandă necesară și capacitatea ruterului vor fi deja pregătite. Pentru o subrețea bazată pe datagrame, evitarea congestionării este mult mai dificilă.

Pentru sistemele de prelucrare a tranzacțiilor (de exemplu apelurile magazinelor pentru a verifica cumpărături realizate cu cărți de credit) overhead-ul implicat de stabilirea și eliberarea unui circuit virtual poate reduce cu ușurință utilitatea circuitului. Dacă majoritatea traficului este de acest tip, folosirea internă a circuitelor virtuale în cadrul subrețelei nu prea are sens. Pe de altă parte, ar putea fi de folos circuite virtuale permanente, stabilite manual și care să dureze luni sau chiar ani.

Circuitele virtuale au o problemă de vulnerabilitate. Dacă un ruter se defectează și își pierde conținutul memoriei, atunci toate circuitele virtuale care treceau prin el sunt suprimate, chiar dacă acesta își revine după o secundă. Prin contrast, dacă se defectează un ruter bazat pe datagrame vor fi afectați doar acei utilizatori care aveau pachete memorate temporar în cozile de așteptare ale ruterului și este posibil ca numărul lor să fie și mai mic, în funcție de câte pachete au fost deja confirmate. Pierderea liniei de comunicație este fatală pentru circuitele virtuale care o folosesc, însă poate fi ușor compensată dacă se folosesc datagrame. De asemenea, datagramele permit ruterului să echilibreze traficul prin subrețea, deoarece căile pot fi modificate parțial în cursul unei secvențe lungi de pachete transmise.

5.2 ALGORITMI DE DIRIJARE

Principala funcție a nivelului rețea este dirijarea pachetelor de la mașina sursă către mașina destinație. În majoritatea subrețelelor pachetele vor face salturi multiple pentru a ajunge la destinație. Singura excepție remarcabilă o reprezintă rețelele cu difuzare, dar chiar și aici dirijarea este importantă, atunci când sursa și destinația nu sunt în aceeași rețea. Algoritmii care aleg calea și structurile de date folosite de aceștia reprezintă un domeniu important al proiectării nivelului rețea.

Algoritmul de dirijare (routing algorithm) este acea parte a software-ului nivelului rețea care răspunde de alegerea liniei de ieșire pe care trebuie trimis un pachet recepționat. Dacă subrețeaua folosește intern datagrame, această decizie trebuie luată din nou pentru fiecare pachet recepționat, deoarece este posibil ca cea mai bună rută să se fi modificat între timp. Dacă subrețeaua folosește circuite virtuale, deciziile de dirijare sunt luate doar la inițializarea unui nou circuit virtual. După aceea pachetele de date vor urma doar calea stabilită anterior. Acest ultim caz este numit uneori **dirijare de sesiune** (**session routing**), deoarece calea rămâne în funcțiune pentru o întreagă sesiune utilizator (de exemplu o sesiune de conectare de la un terminal -login- sau un transfer de fișiere).

Uneori este util să se facă distincția între dirijare, care înseamnă alegerea căii care va fi folosită, și retransmitere, care se referă la ceea ce se întâmplă atunci când sosește un pachet. Se poate spune despre un ruter că rulează intern două procese. Unul dintre ele preia fiecare pachet care sosește, căutând în tabela de dirijare linia de ieșire folosită pentru el. Acesta este procesul de **retransmitere** (**forwarding**). Celălalt proces se ocupă de completarea și actualizarea tabelei de rutare. Aici algoritmul intervine de dirijare.

Indiferent dacă ruta se alege independent pentru fiecare pachet sau doar la stabilirea unei noi conexiuni, un algoritm de dirijare trebuie să aibă anumite proprietăți: corectitudine, simplitate, robustețe, stabilitate, echitate, optimalitate. Corectitudinea și simplitatea nu mai au nevoie de comentarii, dar necesitatea robusteții poate fi mai puțin evidentă la prima vedere. Odată ce apare pe piață o rețea importantă, este de așteptat ca ea să funcționeze continuu ani întregi, fără defecte generale ale sistemului. În acest timp vor exista defecte hardware și software de tot felul. Calculatoare gazdă, rutere, linii de comunicație vor cădea repetat și topologia se va schimba de multe ori. Algoritmul de dirijare trebuie să facă față acestor modificări ale topologiei și traficului, fără a impune ca toate joburile de pe toate calculatoarele să fie abandonate și rețeaua să fie reinițializată de fiecare dată când se defectează un ruter.

Stabilitatea este de asemenea un obiectiv important pentru algoritmul de dirijare. Există algoritmi de dirijare care niciodată nu converg la echilibru, indiferent cât timp ar rula. Un algoritm stabil atinge starea de echilibru și o menține. Echitatea și optimalitatea sunt evidente – este sigur că nici o persoană înțeleaptă nu li se opune - însă, așa cum se va arăta, adeseori acestea sunt obiective contradictorii. Un exemplu simplu al acestui conflict este prezentat în fig. 5-5. Presupunem că între A și A', între B și B' și între C și C' există un trafic suficient pentru a satura legăturile orizontale. Pentru a maximiza fluxul total, traficul între X și X' trebuie oprit. Din păcate acest lucru ar defavoriza pe X și X'. Evident, este necesar un compromis între eficiența globală și echitatea față de fiecare dintre conexiuni.

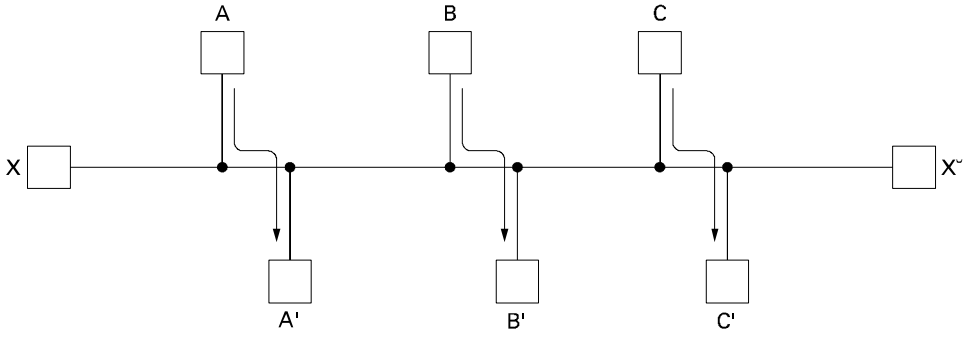


Fig. 5-5. Conflict între echitate și optimalitate.

Înainte de a încerca să găsim rezolvarea acestui conflict între optimalitate și prevenirea defavorizării, trebuie să stabilim ce vrem să optimizăm. Minimizarea întârzierii medii a unui pachet este un candidat evident, însă la fel este și maximizarea productivității (throughput) totale a rețelei. Mai mult, și aceste două obiective sunt în conflict, deoarece funcționarea unui sistem cu cozi de așteptare la limita capacității sale produce întârzieri majore. Pentru a realiza un compromis, în multe rețele se încearcă minimizarea numărului de salturi pe care trebuie să le facă un pachet, deoarece reducerea numărului de salturi tinde să îmbunătățească întârzierea și de asemenea să reducă lățimea de bandă consumată, ceea ce tinde să îmbunătățească și productivitatea.

Algoritmii de dirijare pot fi grupați în două mari clase: neadaptivi și adaptivi. **Algoritmii neadaptivi** (**nonadaptive algorithms**) nu își bazează deciziile de dirijare pe măsurători sau estimări ale traficului și topologiei curente. Astfel, alegerea căii folosite pentru a ajunge de la nodul *I* la nodul *J* (oricare ar fi *I* și *J*) se calculează în avans, off-line și parvine ruterului la inițializarea rețelei. Această procedură se mai numește și **dirijare statică** (**static routing**).

Algoritmii adaptivi (**adaptive algorithms**), prin contrast, își modifică deciziile de dirijare pentru a reflecta modificările de topologie și de multe ori și pe cele de trafic. Algoritmii adaptivi diferă prin locul de unde își iau informația (de exemplu local, de la un ruter vecin sau de la toate ruterele), prin momentul la care schimbă rutele (de exemplu la fiecare ΔT secunde, când se schimbă încărcarea sau când se schimbă topologia) și prin metrica folosită pentru optimizare (de exemplu distanța, numărul de salturi sau timpul estimat pentru tranzit). În secțiunile următoare vom discuta o varietate de algoritmi de dirijare, atât statici cât și dinamici.

5.2.1 Principiul optimalității

Înainte de a intra în algoritmii specifici, ar fi poate folositor să observăm că se poate face o afirmație despre rutele optimale fără a ne referi la topologia rețelei sau la trafic. Această afirmație este cunoscută sub numele de **principiul optimalității** (**optimality principle**). El stabilește că dacă ruterul *J* este pe calea optimă de la ruterul *I* către ruterul *K*, atunci calea optimă de la *J* la *K* este pe aceeași rută. Pentru a vedea aceasta, să notăm cu r_1 partea din cale de la *I* la *J*, iar cu r_2 restul rutei. Dacă ar exista o rută mai bună decât r_2 de la *J* la *K*, ea ar putea fi concatenată cu r_1 și ar îmbunătăți ruta de la *I* la *K*, ceea ce ar contrazice presupunerea că r_1r_2 este optimală.

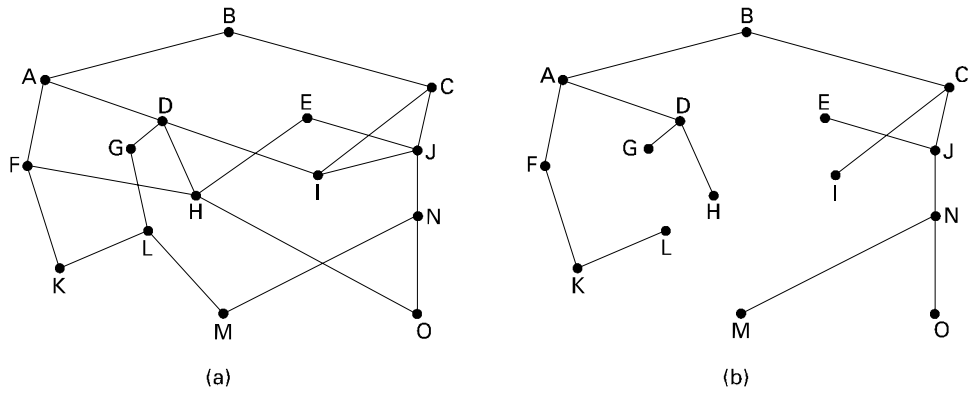


Fig. 5-6. (a) O subrețea. (b) Un arbore de scufundare pentru ruterul *B*.

Ca o consecință directă a principiului optimalității, putem observa că mulțimea rutelor de la toate sursele către o anume destinație formează un arbore având rădăcina în destinație. Acest arbore se numește **arbore de scufundare** (**sink tree**) și este prezentat în fig. 5-6, unde distanța metrica aleasă este numărul de salturi. Observați că arborele de scufundare nu este unic, putând exista și alți arbori cu aceeași lungime a căii. Scopul tuturor algoritmilor de dirijare este de a descoperi și folosi arborii de scufundare pentru toate ruterele.

Deoarece arborele de scufundare este într-adevăr un arbore, el nu conține bucle, deci fiecare pachet va fi livrat într-un număr finit și limitat de salturi. În practică viața nu este chiar așa de ușoară. Legăturile și ruterele pot să se defecteze și să-și revină în timpul operațiilor, astfel încât diferite rutere pot avea imagini diferite asupra topologiei curente. De asemenea, am trecut mai repede peste întrebarea dacă fiecare ruter trebuie să obțină individual informația necesară calculării arborelui de scufundare sau dacă această informație este colectată prin alte mijloace. Vom reveni însă la această

problemă în curând. Cu toate acestea, principiul optimalității și arborele de scufundare furnizează referințe cu care pot fi comparați ceilalți algoritmi de dirijare.

5.2.2 Dirijarea pe calea cea mai scurtă

Să începem studiul algoritmilor de dirijare posibili cu o tehnică des utilizată în multe forme deoarece este simplă și ușor de înțeles. Ideea este de a construi un graf al subrețelei, fiecare nod al grafului fiind un ruter, iar fiecare arc al grafului fiind o linie de comunicație (numită adesea legătură). Pentru a alege o cale între o pereche dată de rutere, algoritmul trebuie să găsească în graf calea cea mai scurtă dintre ele.

Conceptul de **cea mai scurtă cale (shortest path routing)** necesită unele explicații. O modalitate de a măsura lungimea căii este numărul de salturi. Folosind această metrică, căile *ABC* și *ABE* din fig. 5-7 sunt la fel de lungi. O altă metrică este distanța geografică în kilometri, caz în care *ABC* este clar mult mai mare decât *ABE* (presupunând că fig. este desenată la scară).

Oricum, sunt posibile multe alte metrici în afară de salturi și distanța geografică. De exemplu, fiecare arc poate fi etichetat cu valorile medii ale așteptării în coadă și întârzierii de transmisie pentru anumite pachete standard de test, așa cum sunt determinate de măsurători care se fac din oră în oră. Cu această etichetare, cea mai scurtă cale este cea mai rapidă, nu neapărat cea cu mai puține arce sau kilometri.

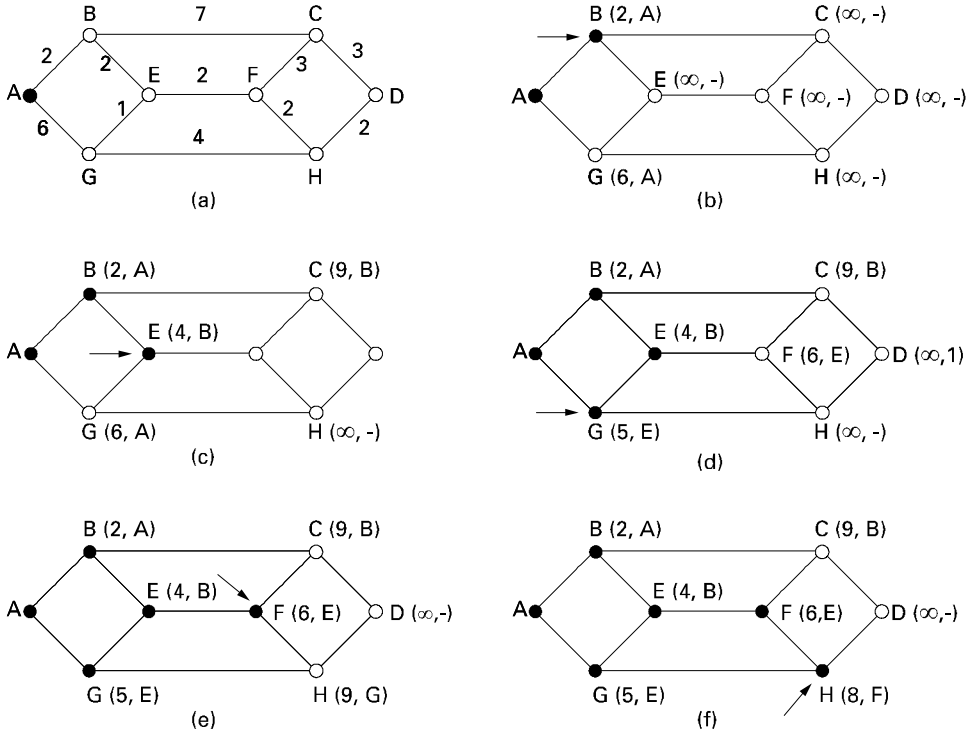


Fig. 5-7. Primii cinci pași folosiți în calcularea celei mai scurte căi de la A la D. Săgețile indică nodul curent.

În cazul cel mai general, etichetele de pe arce ar putea fi calculate ca funcții de distanță, lărgime de bandă, trafic mediu, cost al comunicației, lungime medie a cozilor de așteptare, întârzieri măsurate și alți factori. Prin modificarea ponderilor, algoritmul ar putea calcula cea mai „scurtă" cale, în conformitate cu oricare dintre aceste criterii sau cu combinații ale acestor criterii.

Se cunosc mai mulți algoritmi pentru calculul celei mai scurte căi între două noduri dintr-un graf. Cel mai cunoscut este cel propus de Dijkstra (1959). Fiecare nod este etichetat (în paranteze) cu distanța de la nodul sursă până la el, de-a lungul celei mai bune căi cunoscute. Inițial nu se cunoaște nici o cale, așa că toate nodurile vor fi etichetate cu infinit. Pe măsură ce se execută algoritmul și se găsesc noi căi, etichetele se pot schimba, reflectând căi mai bune. O etichetă poate fi fie temporară, fie permanentă. Inițial toate etichetele sunt temporare. Atunci când se descoperă că o etichetă reprezintă cea mai scurtă cale posibilă de la sursă către acel nod, ea devine permanentă și nu se mai schimbă ulterior.

Pentru a ilustra cum funcționează algoritmul de etichetare, să ne uităm la graful neorientat, etichetat din fig. 5-7(a), unde etichetele reprezintă, de exemplu, distanța. Dorim să aflăm cea mai scurtă cale de la *A* la *D*. Începem prin a marca nodul *A* ca permanent, indicând aceasta printr-un cerc colorat. Apoi vom examina fiecare nod adiacent cu *A* (care este acum nodul curent), reetichetând fiecare nod cu distanța până la nodul *A*. De fiecare dată când un nod este reetichetat, îl vom eticheta și cu nodul de la care s-a făcut încercarea, pentru a putea reface calea ulterior. După ce am examinat toate nodurile adiacente ale lui *A*, vom examina toate nodurile cu etichetă temporară din întregul graf și îl facem permanent pe cel cu eticheta minimă, așa cum se observă din fig. 5-7(b). Acest nod devine noul nod curent.

Acum începem din *B* și examinăm toate nodurile sale adiacente. Dacă suma între eticheta lui *B* și distanța de la *B* la nodul considerat este mai mică decât eticheta acelui nod, înseamnă că am găsit o cale mai scurtă și va trebui făcută reetichetarea nodului.

După ce toate nodurile adiacente nodului curent au fost inspectate și au fost schimbate toate etichetele temporare posibile, se reia căutarea în întregul graf pentru a identifica nodul cu eticheta temporară minimă. Acest nod este făcut permanent și devine nodul curent al etapei următoare. Fig. 5-7 prezintă primii cinci pași ai algoritmului.

Pentru a vedea de ce merge algoritmul, să privim fig. 5-7(c). La momentul respectiv de abia am făcut permanent nodul *E*. Să presupunem că ar exista o cale mai scurtă decât *ABE*, de exemplu *AXYZE*. Există două posibilități: fie nodul *Z* a fost deja făcut permanent, fie încă nu a fost. Dacă a fost, atunci *E* a fost deja examinat (la pasul imediat următor celui la care *Z* a fost făcut permanent), astfel încât calea *AXYZE* nu a fost ignorată și deci nu poate fi cea mai scurtă cale.

Să considerăm acum cazul în care *Z* este încă etichetă temporară. Atunci fie eticheta lui *Z* este mai mare sau egală cu cea a lui *E*, caz în care *ABE* nu poate fi o cale mai scurtă decât *AXYZE*, fie este mai mică decât cea a lui *E*, caz în care *Z* și nu *E* va deveni permanent mai întâi, permițând lui *E* să fie examinat din *Z*.

Algoritmul este prezentat în fig. 5-8. Variabilele globale *n* și *dist* sunt inițializate înainte să fie apelată *shortest_path*. Singura diferență între program și algoritmul descris mai sus este aceea că în fig. 5-8 calculăm calea cea mai scurtă pornind de la nodul terminal, *t*, în locul nodului sursă, *s*. Deoarece calea cea mai scurtă de la *t* la *s* într-un graf neorientat este exact aceeași cu calea cea mai scurtă de la *s* la *t*, nu contează la care capăt începem (decât dacă există mai multe căi scurte, caz în care, inversând calea, am putea găsi alt drum). Motivul pentru care începem căutarea de la nodul destinație este acela că nodurile sunt etichetate cu predecesorul și nu cu succesorul. Atunci când calea finală este copiată în variabila de ieșire, *path*, calea este inversată. Prin inversarea căutării, cele două efecte se anulează, astfel încât rezultatul se obține în ordinea corectă.

```
#define MAX_NODES 1024                                  /* numărul maxim de noduri */
#define INFINITY 1000000000                          /* un număr mai mare decât orice cale */
int n,dist[MAX_NODES][ MAX_NODES]                    /* dist[i][j] e distanța de la i la j */

void shortest_path(int s, int t, int path[])
{ struct state {                                          /* calea cu care se lucrează */
 int predecessor;                                                 /* nodul anterior */
 int length;                                          /* lungimea de la sursă la acest nod */
 enum {permanent, tentative} label;                               /* etichetă stare */
}state [MAX_NODES];

int i, k, min;
struct state *p;

for (p = &state[0]; p < &state[n]; p++) {                          /* inițializări */
 p->predecessor = -1;
 p->length = INFINITY;
 p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                                                  /* k este nodul inițial de lucru */
do {                                                 /* există vreo cale mai bună de la k? */
 for (i = 0; i < n; i++)                                          /* graful are n noduri */
   if (dist[k][i] != 0 && state[i].label == tentative) {
     if (state[k].length + dist[k][i] < state[i].length) {
       state[i].predecessor = k;
       state[i].length = state[k].length + dist[k][i];
     }
   }

 /* Găsește nodul etichetat temporar cu cea mai mică etichetă */
 k = 0; min = INFINITY;
 for (i = 0; i < n; i++)
   if (state[i].label == tentative && state[i].length < min) {
     min = state[i].length;
     k = i;
   }
 state[k].label = permanent;
} while (k != s);

/* Copiază calea în vectorul de ieșire */
i = 0; k = s;
do { path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Fig. 5-8. Algoritmul Dijkstra pentru calculul celei mai scurte căi într-un graf.

5.2.3 Inundarea

Un alt algoritm static este **inundarea (flooding)**, în care fiecare pachet recepționat este trimis mai departe pe fiecare linie de ieșire, cu excepția celei pe care a sosit. Este evident că inundarea generează un mare număr de pachete duplicate, de fapt un număr infinit dacă nu se iau unele măsuri pentru a limita acest proces. O astfel de măsură este păstrarea unui contor de salturi în antetul fiecă-

rui pachet, contor care este decrementat la fiecare salt și care face ca pachetul să fie distrus când contorul atinge valoarea zero.

Ideal ar fi ca acest contor să fie inițializat cu lungimea căii de la sursă la destinație. Dacă emițătorul nu cunoaște lungimea căii, poate inițializa contorul la valoarea cea mai defavorabilă, adică diametrul subrețelei.

O metodă alternativă pentru limitarea inundării este identificarea pachetelor care au fost deja inundate, pentru a preîntâmpina trimiterea lor a doua oară. O cale pentru a realiza acest scop este ca ruterul sursă să plaseze un număr de secvență în fiecare pachet pe care îl primește de la calculatorul gazdă asociat. Fiecare ruter necesită menținerea unei liste pentru fiecare ruter sursă, cu numerele de secvență provenite de la acel ruter sursă și care au fost deja trimise mai departe. Dacă sosește un pachet care se află în listă, el nu mai este trimis mai departe.

Pentru a limita creșterea lungimii listei, fiecare listă trebuie însoțită de un contor, *k*, care semnifică faptul că toate numerele de secvență până la *k* au fost deja tratate. La recepția unui pachet este ușor să se verifice dacă este un duplicat, caz în care este distrus. Evident, lista cu numere mai mici decât *k* nu este necesară, deoarece *k* o rezumă.

O variantă a algoritmului de inundare, care este și ceva mai practică, este **inundarea selectivă** (**selective flooding**). În acest algoritm ruterele nu trimit fiecare pachet recepționat pe fiecare legătură de ieșire, ci doar pe acele linii care duc aproximativ în direcția potrivită. De obicei sunt puține motive pentru a trimite un pachet spre partea de vest a rețelei folosind o legătură spre est, decât dacă topologia rețelei este cu totul deosebită și ruterul este sigur de acest lucru.

Inundarea nu este practică pentru majoritatea aplicațiilor, însă are destule utilizări. De exemplu, în aplicațiile militare, unde un mare număr de rutere pot fi scoase din funcționare în orice moment, robustețea extraordinară a inundării este necesară. În aplicațiile de baze de date distribuite, este uneori necesar ca toate bazele de date să fie actualizate simultan, caz în care inundarea poate fi folositoare. În rețelele fără fir, toate mesajele transmise de o gazdă pot fi recepționate de toate celelalte gazde din raza sa radio, ceea ce înseamnă de fapt inundare, și unii algoritmi folosesc această proprietate. O a patra utilizare posibilă a inundării este ca metrică la care să se raporteze toți ceilalți algoritmi de dirijare. Inundarea alege întotdeauna cea mai scurtă cale, deoarece alege în paralel toate căile posibile. În consecință, nici un alt algoritm nu poate produce o întârziere mai redusă (dacă ignorăm supraîncărcarea generată de însuși procesul de inundare).

5.2.4 Dirijare cu vectori distanță

Rețelele moderne de calculatoare folosesc de obicei algoritmi dinamici de dirijare în locul celor statici, descriși anterior, deoarece algoritmii statici nu țin seama de încărcarea curentă a rețelei. Doi dintre cei mai cunoscuți algoritmi dinamici sunt algoritmul de dirijare cu vectori distanță și algoritmul de dirijare bazat pe starea legăturilor. În această secțiune ne vom ocupa de primul algoritm. În secțiunea următoare vom studia cel de-al doilea algoritm.

Algoritmul de **dirijare cu vectori distanță (distance vector routing)** presupune că fiecare ruter menține o tabelă (de exemplu un vector) care păstrează cea mai bună distanță cunoscută spre fiecare destinație și linia care trebuie urmată pentru a ajunge acolo. Aceste tabele sunt actualizate prin schimbul de informații între nodurile vecine.

Algoritmul de dirijare cu vectori distanță este cunoscut și sub alte nume, cel mai des algoritmul distribuit de dirijare **Bellman-Ford** și algoritmul **Ford-Fulkerson**, după numele cercetătorilor care l-

au propus (Bellman, 1957; și Ford și Fulkerson, 1962). A fost algoritmul de dirijare folosit inițial în rețeaua ARPANET, a fost folosit de asemenea în Internet sub numele de RIP.

În dirijarea pe baza vectorilor distanță, fiecare ruter păstrează o tabelă de dirijare conținând câte o intrare pentru fiecare ruter din subrețea. Această intrare are două părți: linia de ieșire preferată care se folosește pentru destinația respectivă și o estimare a timpului sau distanței până la acea destinație. Metrica folosită poate fi numărul de salturi, întârzierea în milisecunde, numărul total de pachete care așteaptă în cozi de-a lungul căii, sau ceva asemănător.

Se presupune că ruterul cunoaște „distanța" spre fiecare dintre vecinii săi. Dacă se folosește metrica salturilor, distanța este doar de un salt. Dacă metrica folosită este cea a lungimii cozilor de așteptare, ruterul examinează pur și simplu lungimile acestor cozi. Dacă metrica este cea a întârzierilor, ruterul o poate măsura direct prin pachete speciale ECHO, în care receptorul va marca doar timpul curent (ștampila de timp) și le va trimite înapoi cât mai repede posibil.

Ca exemplu, să presupunem că se folosește metrica întârzierilor și că ruterul cunoaște întârzierea spre fiecare dintre vecinii săi. O dată la fiecare T msec fiecare ruter trimite spre fiecare vecin o listă a estimărilor proprii spre fiecare destinație. De asemenea el recepționează o listă similară de la fiecare vecin. Să presupunem că una dintre aceste tabele tocmai a sosit de la vecinul X, cu X_i fiind estimarea lui X despre cât timp este necesar pentru a ajunge la ruterul i. Dacă ruterul știe că întârzierea spre X este m msec, el știe de asemenea că poate atinge ruterul i trecând prin X în $X_i + m$ msec. Făcând aceste calcule pentru fiecare vecin, un ruter poate stabili care estimare pare a fi cea mai bună, pentru a folosi această estimare, împreună cu linia corespunzătoare în noua tabelă de dirijare. Este de remarcat faptul că vechea tabelă de dirijare nu intervine în calcule.

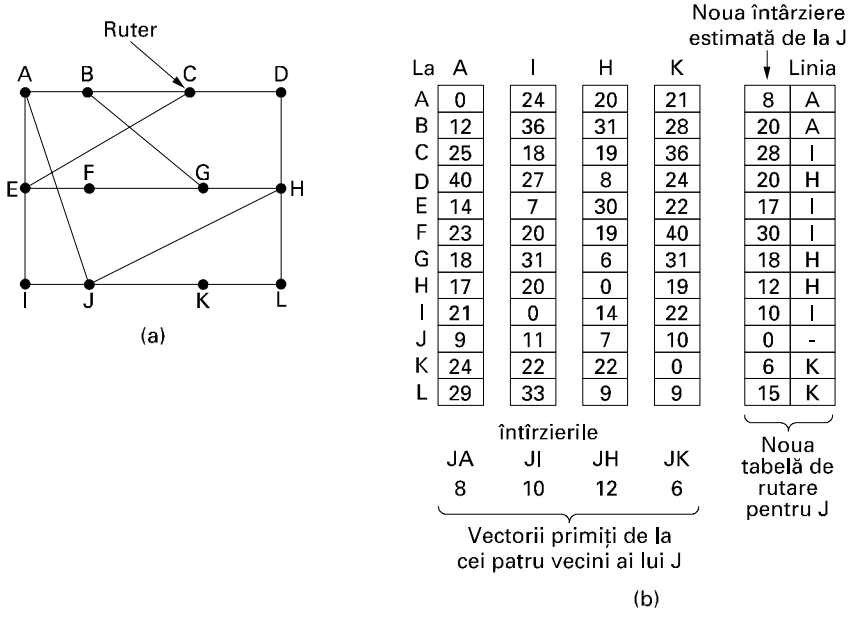


Fig. 5-9. (a) O subrețea. (b) Intrări de la A, I, H și K și noua tabelă de dirijare pentru J.

Acest proces de actualizare este ilustrat în fig. 5-9. Partea (a) prezintă o subrețea. Primele patru coloane din partea (b) conțin vectorii de întârzieri primiți de la vecinii ruterului J. A afirmă că are 12 msec întârziere spre B, 25 msec întârziere spre C, 40 msec întârziere spre D etc. Presupunem că

J și-a măsurat sau estimat întârzierea față de vecinii săi *A*, *I*, *H* și *K*, obținând valorile 8, 10, 12 și 16 ms, respectiv.

Să vedem cum calculează *J* noua cale spre ruterul *G*. El știe că poate ajunge la *A* în 8 msec și *A* pretinde că este în stare să ajungă la *G* în 18 msec, astfel încât *J* poate conta pe o întârziere de 26 msec spre G dacă dirijează pachetul spre *A*. Similar, el calculează întârzierea spre *G* prin *I*, *H*, *K* ca fiind 41 (31 + 10), 18 (6 + 12) și 37 (31 + 6) respectiv. Cea mai bună valoare este 18, așa că va crea o intrare în tabela de dirijare cu întârzierea către *G* de 18 msec și ruta de urmat trecând prin *H*. Aceleași calcule se fac pentru toate destinațiile, obținându-se noua tabelă de dirijare, care este prezentată în ultima coloană a figurii.

Problema numărării la infinit

Dirijarea folosind vectori distanță funcționează în teorie, însă în practică are o limitare importantă: deși ea converge spre rezultatul corect, o face foarte lent. În particular, ea reacționează rapid la veștile bune, dar foarte lent la cele rele. Să considerăm un ruter care are un cel mai bun drum spre destinația *X* foarte lung. Dacă la următorul schimb de informații, vecinul său *A* raportează brusc o întârziere mică spre *X*, ruterul va comuta și va folosi linia spre *A* pentru a dirija traficul spre *X*. Astfel, într-o singură schimbare a vectorului, vestea bună a fost luată în considerare.

Pentru a vedea cât de repede se propagă veștile bune, să considerăm subrețeaua (liniară) de cinci noduri din fig. 5-10, unde metrica întârzierilor este numărul de salturi. Să presupunem că inițial nodul *A* nu funcționează și toate celelalte rutere cunosc acest lucru. Cu alte cuvinte, toate celelalte rutere au înregistrat întârzierea spre *A* ca având valoarea infinit.

Când *A* pornește, celelalte rutere află aceasta datorită schimbărilor din vector. Pentru simplificare, vom considera că există un gong uriaș undeva, care bate periodic pentru a iniția schimbul de vectori simultan la toate ruterele. La momentul primului schimb, *B* află că vecinul din stânga are o întârziere nulă spre *A*. Astfel, *B* creează o nouă intrare în tabela sa, marcând faptul că *A* este la un singur salt distanță, spre stânga. Toate celelalte rutere consideră că *A* este încă oprit. Intrările tabelei de dirijare pentru *A*, la acest moment, sunt prezentate în a doua linie din fig. 5-10(a). La următorul schimb, *C* află că *B* are o cale de lungime 1 spre *A*, astfel încât își actualizează tabela de dirijare pentru a indica o cale de lungime 2, însă *D* și *E* nu vor primi vestea cea bună decât mai târziu. Evident, vestea cea bună se răspândește cu viteza de un salt la fiecare schimb. Într-o subrețea având calea cea mai lungă de lungime *N* salturi, după *N* schimburi fiecare ruter va afla despre liniile și ruterele nou apărute.

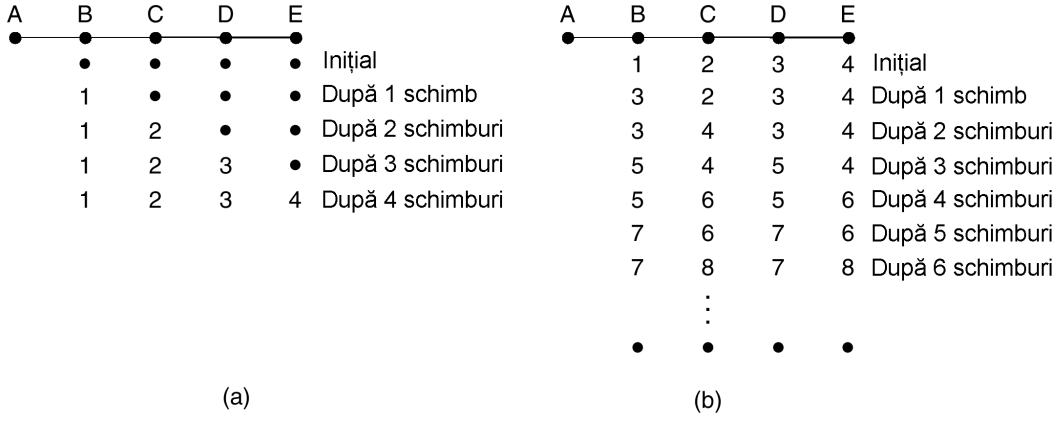

A B C D E
Inițial
1 ● ● ● După 1 schimb
1 2 ● ● După 2 schimburi
1 2 3 ● După 3 schimburi
1 2 3 4 După 4 schimburi
(a)
A B C D E
1 2 3 4 Inițial
3 2 3 4 După 1 schimb
3 4 3 4 După 2 schimburi
5 4 5 4 După 3 schimburi
5 6 5 6 După 4 schimburi
7 6 7 6 După 5 schimburi
7 8 7 8 După 6 schimburi
(b)

Fig. 5-10. Problema numărării la infinit.

Să considerăm acum situația din fig. 5-10(b), în care toate liniile și ruterele sunt inițial în funcțiune. Ruterele *B*, *C*, *D* și *E* au distanțele spre *A* respectiv de 1, 2, 3, 4. Brusc, *A* se oprește sau, alternativ, linia dintre *A* și *B* este întreruptă, ceea ce reprezintă efectiv același lucru din punctul de vedere al lui *B*.

La primul schimb de pachete, *B* nu primește nimic de la *A*. Din fericire, *C* spune: „Nici o problemă. Eu știu o cale spre *A* de lungime 2." Însă *B* nu știe că această cale a lui *C* trece prin *B* însuși. După cunoștințele lui *B*, *C* ar putea avea zece linii, toate cu căi separate de lungime 2 spre *A*. Prin urmare *B* va crede că poate ajunge la *A* prin *C* pe o cale de lungime 3. *D* și *E* nu își actualizează intrările proprii pentru *A* la primul schimb.

La al doilea schimb, *C* remarcă faptul că fiecare dintre vecinii săi pretinde a avea o cale de lungime 3 spre *A*. El va alege la întâmplare unul dintre acești vecini și va înregistra noua distanță spre *A* ca fiind 4, așa cum se arată în linia a treia din fig. 5-10(b). Schimburile următoare vor produce succesiunea prezentată în continuare în fig. 5-10(b).

Din această figură se poate deduce de ce veștile rele circulă mai lent: orice ruter va avea întotdeauna o valoare cu cel mult unu mai mare decât valoarea minimă a vecinilor săi. Treptat, toate ruterele vor ajunge la infinit, însă numărul de schimburi necesar depinde de valoarea numerică folosită pentru a reprezenta valoarea infinit. Din această cauză este recomandat să se aleagă infinitul, ca fiind lungimea celei mai mari căi, plus 1. Dacă metrica este întârzierea în timp, atunci nu este definită nici o limită superioară, astfel încât este necesară o valoare mare pentru a preveni considerarea unui drum cu întârziere mare ca fiind un drum defect. Nu este deloc surprinzător că această problemă este numită **problema numărării la infinit** (**the count to infinity problem**). Au existat câteva încercări de rezolvare a problemei (cum ar fi orizont împărțit cu revers otrăvit - eng.: split horizon with poisoned reverse - în RFC 1058), dar nici una nu a funcționat bine în general. Miezul problemei este că atunci când *X* îi spune lui *Y* că are o cale spre o destinație, *Y* nu are de unde să știe dacă se află el însuși pe acea cale.

5.2.5 Dirijarea folosind starea legăturilor

Dirijarea folosind vectori distanță a fost folosită în ARPANET până în 1979, când a fost înlocuită prin dirijarea folosind starea legăturilor. Au fost două probleme importante care au cauzat această schimbare. În primul rând, deoarece metrica folosită era lungimea cozilor de așteptare, la stabilirea rutei nu se lua în considerare lățimea de bandă. Inițial toate liniile erau de 56 Kbps, astfel încât lățimea de bandă nu era o problemă, însă după ce câteva linii au fost îmbunătățite la 230 Kbps, iar altele la 1.544 Mbps, neluarea în considerare a lățimii de bandă a devenit o problemă majoră. Evident, era posibil să se schimbe metrica folosită pentru a depinde și de lățimea de bandă, însă exista și o a doua problemă și anume aceea că algoritmul convergea destul de greu (problema numărării la infinit). De aceea, a fost înlocuit cu un algoritm nou, numit algoritm de **dirijare folosind starea legăturilor (link state routing)**. Variante de dirijare folosind starea legăturilor sunt actualmente foarte răspândite.

Ideea algoritmului bazat pe starea legăturilor este simplă și poate fi formulată în 5 puncte. Fiecare ruter trebuie să facă următoarele:

1. Să descopere care sunt vecinii săi și afle adresele de rețea ale acestora.
2. Să măsoare întârzierea sau costul până la fiecare din vecinii săi.
3. Să pregătească un pachet prin care anunță pe toată lumea că tocmai a terminat de cules datele despre vecini.

4. Să trimită acest pachet către toate celelalte rutere.
5. Să calculeze cea mai scurtă cale spre fiecare ruter.

Ca urmare, întreaga topologie și toate întârzierile sunt măsurate experimental și distribuite spre fiecare ruter. Apoi se poate rula algoritmul lui Dijkstra pentru a afla cea mai scurtă cale către fiecare ruter. În continuare vom analiza mai în detaliu acești cinci pași.

Determinarea vecinilor

Când un ruter este pus în funcțiune, prima sa sarcină este să afle care sunt vecinii săi. El realizează aceasta prin trimiterea unui pachet special HELLO pe fiecare linie prin care este legat la alt ruter. Ruterul de la celălalt capăt trebuie să răspundă anunțând într-un pachet identitatea sa. Aceste nume trebuie să fie unice global, pentru că dacă mai târziu un ruter află că trei rutere sunt conectate toate la *F*, este esențial ca acesta să poată determina dacă cele trei se referă la același *F*.

Când două sau mai multe rutere sunt conectate printr-o rețea LAN, situația devine puțin mai complicată. Fig. 5-11(a) ilustrează un LAN la care sunt conectate direct ruterele *A*, *C* și *F*. Fiecare dintre aceste rutere este conectat cu unul sau mai multe alte rutere, așa cum se arată în figură.

O modalitate de a modela rețeaua LAN este de a o considera ca un nod, așa cum se arată în fig. 5-11 (b). Aici am introdus un nod nou, artificial, *N*, la care *A*, *C* și *F* sunt conectate. Faptul că este posibil să se meargă de la *A* la *C* prin LAN este reprezentat aici de calea *ANC*.

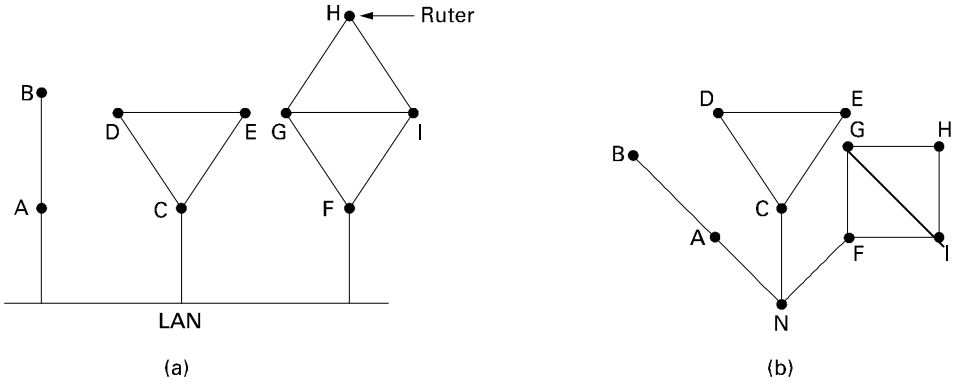


Fig. 5-11. (a) Nouă rutere și o LAN. (b) Graful asociat punctului (a).

Măsurarea costului liniei

Algoritmul de dirijare bazat pe starea legăturilor cere ca fiecare ruter să știe, sau cel puțin să aibă o estimare rezonabilă, a întârzierii către fiecare dintre vecinii săi. Cel mai direct mod de a afla acest lucru este de a trimite un pachet special ECHO pe linie, cerând ca ruterul partener să-l trimită înapoi imediat. Măsurând timpul în care pachetul se întoarce (round-trip time) și împărțindu-l la doi, ruterul inițiator poate avea o estimare rezonabilă a întârzierii. Pentru rezultate și mai bune, testul poate fi repetat de mai multe ori, folosindu-se apoi valoarea medie obținută. Bineînțeles, această metodă presupune implicit că întârzierile sunt simetrice, dar nu mereu este așa.

O problemă interesantă este dacă să se considere sau nu încărcarea rețelei la măsurarea întârzierii. Pentru a ține cont de încărcare, timpul de revenire trebuie măsurat din momentul în care pachetul ECHO este pus în coadă. Pentru a ignora încărcarea, ceasul se poate porni în momentul în care pachetul ECHO ajunge pe prima poziție din coadă.

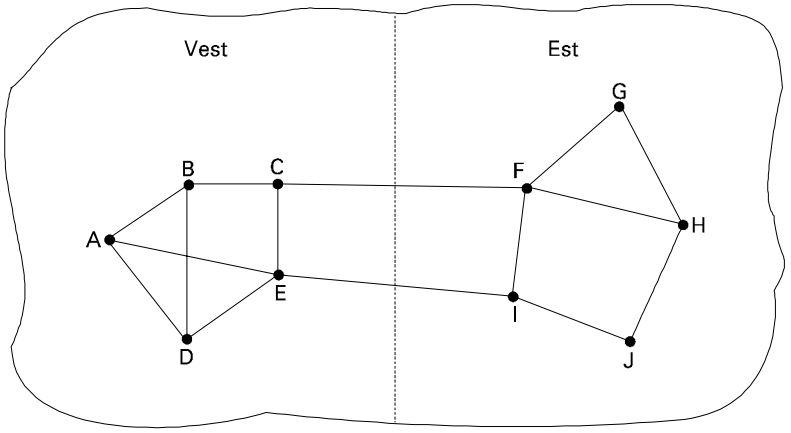


Fig. 5-12. O subrețea în care părțile de Est și Vest sunt conectate prin două linii.

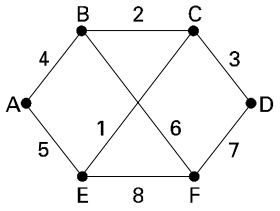
Pot fi aduse argumente în favoarea ambelor variante. Dacă se ține cont de întârzierile provocate de trafic la măsurători înseamnă că dacă un ruter trebuie să aleagă între două linii cu aceeași lățime de bandă, una dintre ele fiind puternic încărcată tot timpul, iar cealaltă nefiind foarte încărcată, atunci ruterul va desemna calea cea mai puțin încărcată ca fiind cea mai scurtă. Această alegere va duce la îmbunătățirea performanțelor.

Din păcate, există și un argument împotriva folosirii încărcării la calculul întârzierii. Să considerăm subrețeaua din fig. 5-12, care este divizată în două zone, Est și Vest, conectate prin două linii, *CF* și *EI*. Să presupunem că majoritatea traficului între Est și Vest folosește linia *CF* și, prin urmare, această linie este puternic încărcată și are întârzieri mari. Folosirea întârzierilor în cozi pentru calculul celei mai scurte căi va face ca drumul *EI* să fie preferat. După ce noile tabele de dirijare au fost instalate, majoritatea traficului Est-Vest va trece acum prin *EI*, supraîncărcând-o. De aceea, la următoarea actualizare, *CF* va părea a fi calea cea mai scurtă. Prin urmare tabelele de dirijare vor oscila puternic, conducând la o dirijare fluctuantă și facilitând apariția multor probleme potențiale. Dacă nu se ține cont de încărcare, luându-se în considerare doar lățimea de bandă, această problemă nu mai apare. Alternativ, încărcarea poate fi distribuită pe ambele linii, dar această soluție nu folosește în întregime calea cea mai bună. Cu toate acestea, pentru a evita oscilațiile în alegerea celei mai bune căi, poate fi înțelept să se distribuie încărcarea pe mai multe linii, în proporții cunoscute pe fiecare linie.

Construirea pachetelor cu starea legăturilor

De îndată ce a fost colectată informația necesară pentru realizarea schimbului, se poate trece la pasul următor, fiecare ruter construind un pachet care conține toate datele. Pachetul începe cu identitatea expeditorului, urmată de un număr de secvență, vârstă (care va fi descrisă în continuare) și o listă a vecinilor. Pentru fiecare vecin se specifică întârzierea asociată. Un exemplu de subrețea este prezentat în fig. 5-13(a), unde etichetele liniilor reprezintă întârzierile asociate. Pachetele cu starea legăturilor asociate tuturor celor șase rutere sunt prezentate în fig. 5-13(b).

Construirea pachetelor cu starea legăturilor se face ușor. Partea mai dificilă este să se determine când să fie construite ele. O posibilitate este ca ele să fie construite periodic, adică la intervale regulate. O altă posibilitate este atunci când se produce un eveniment semnificativ, cum ar fi scoaterea din funcțiune a unui vecin sau a unei linii, sau repunerea lor în funcțiune, sau modificarea semnificativă a proprietăților lor.



(a)

Pachete cu starea legăturilor

A	
Secv.	
Vârstă	
B	4
E	5

B	
Secv.	
Vârstă	
A	4
C	2
F	6

C	
Secv.	
Vârstă	
B	2
D	3
E	1

D	
Secv.	
Vârstă	
C	3
F	7

E	
Secv.	
Vârstă	
A	5
C	1
F	8

F	
Secv.	
Vârstă	
B	6
D	7
E	8

(b)

Fig. 5-13. (a) O subrețea. (b) Pachetele cu starea legăturilor pentru această subrețea.

Distribuirea pachetelor cu starea legăturilor

Cea mai complicată parte a algoritmului este distribuirea sigură a pachetelor cu starea legăturilor. De îndată ce pachetele sunt distribuite și instalate, ruterele care primesc primele pachete își vor schimba rutele. În consecință, rutere diferite ar putea folosi versiuni diferite ale topologiei, ceea ce poate duce la apariția unor inconsistențe, bucle, mașini inaccesibile sau a altor probleme.

Pentru început vom descrie algoritmul de bază folosit pentru distribuție. Apoi vom prezenta unele îmbunătățiri posibile. Ideea fundamentală este folosirea inundării pentru a distribui pachetele cu starea legăturilor. Pentru a avea controlul inundării, fiecare pachet conține un număr de secvență care este incrementat la fiecare nou pachet trimis. Ruterele păstrează evidența tuturor perechilor (ruter sursă, număr secvență) pe care le văd. La sosirea unui nou pachet cu starea legăturilor, el este căutat în lista pachetelor deja văzute. Dacă pachetul este nou, el este retrimis pe toate liniile, cu excepția celei pe care a sosit. Dacă este un duplicat, pachetul este distrus. Dacă pachetul sosit are un număr de secvență mai mic decât cel mai mare număr de secvență detectat, atunci el este rejectat ca fiind învechit.

Acest algoritm are câteva probleme, dar ele sunt tratabile. În primul rând, dacă numerele de secvență ating valoarea maximă posibilă și reîncep de la valori mici, se pot produce confuzii. Soluția este folosirea numerelor de secvență pe 32 biți. Considerând un pachet de starea legăturilor pe secundă, ar trebui 137 ani pentru a se reîncepe de la valori mici, astfel încât această posibilitate poate fi ignorată.

În al doilea rând, dacă un ruter se defectează, el va pierde evidența numerelor de secvență. Dacă va reîncepe de la 0, următorul pachet va fi rejectat ca duplicat.

În al treilea rând, dacă numărul de secvență este alterat și se recepționează 65540 în loc de 4 (eroare de modificare a unui bit), pachetele de la 5 la 65540 vor fi rejectate ca fiind învechite, deoarece se consideră că numărul de secvență curent este 65540.

Soluția tuturor acestor probleme este includerea vârstei pachetului după numărul de secvență și decrementarea sa la fiecare secundă. Când vârsta ajunge la zero, informația de la ruterul respectiv este distrusă. În mod normal un nou pachet apare, să zicem, la fiecare 10 sec., astfel încât informația de la ruter expiră doar dacă ruterul este oprit (sau dacă șase pachete consecutive s-au pierdut, un lucru extrem de improbabil). Câmpul *Age* este decrementat de asemenea de fiecare ruter la începutul procesului de inundare, pentru a se asigura că nici un pachet nu se poate pierde sau nu poate supraviețui o perioadă nelimitată (un pachet având vârsta zero este distrus).

Unele îmbunătățiri ale algoritmului pot să-l facă mai robust. Atunci când un pachet cu starea legăturilor ajunge într-un ruter pentru inundare, acesta nu este pus imediat în coada de transmisie. El este pus într-o zonă de așteptare pentru o scurtă perioadă. Dacă înainte de a fi trimis primul pachet sosește un alt pachet cu starea legăturilor de la aceeași sursă, numerele lor de secvență sunt comparate. Dacă sunt identice, duplicatul este distrus. Dacă sunt diferite, cel mai bătrân este ignorat. Pen-

tru a preîntâmpina erorile pe linia ruter-ruter, toate aceste pachete sunt confirmate. Dacă linia nu este folosită, zona de așteptare este inspectată în manieră round-robin, pentru a selecta un pachet sau o confirmare de trimis.

Structura de date folosită de ruterul *B* pentru subrețeaua din fig. 5-13(a) este descrisă în fig. 5-46. Fiecare linie corespunde unui pachet cu starea legăturilor sosit recent, dar încă neprelucrat în întregime. În tabelă se înregistrează originea pachetului, numărul de secvență, vârsta și datele transportate. În plus mai există unele indicatoare (flags) de trimitere și confirmare pentru fiecare dintre cele trei linii ale lui *B* (respectiv către *A*, *C*, *F*). Indicatorul de trimitere precizează că pachetul trebuie trimis pe linia indicată. Indicatorul de confirmare precizează că respectivul pachet trebuie confirmat.

În fig. 5-14, pachetul cu starea legăturilor de la *A* sosește direct, astfel încât el trebuie trimis către *C* și *F* și trebuie confirmat către *A*, așa cum precizează biții indicatori. Similar, pachetul de la *F* trebuie trimis către *A* și *C* și confirmat către *F*.

			Trimitere			ACK			
Sursa	Secv.	Vârsta	A	C	F	A	C	F	Date
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Fig. 5-14. Zona tampon pentru pachete a ruterului *B* din fig. 5-13.

Oricum, situația celui de-al treilea pachet, de la *E*, este diferită. El sosește de două ori, întâi pe calea *EAB* și apoi pe calea *EFB*. În consecință el trebuie trimis numai către *C*, însă trebuie confirmat atât lui *A* cât și lui *F*, așa cum indică și biții corespunzători.

Dacă sosește un duplicat în timp ce pachetul este încă în zona tampon, biții trebuie modificați. De exemplu, dacă o copie a stării lui *C* sosește de la *F* înainte ca a patra intrare din tabelă să fie trimisă, cei șase biți se vor schimba în 100011, pentru a preciza că pachetul trebuie confirmat către *F*, dar nu trimis acolo.

Calcularea noilor rute

De îndată ce un ruter a acumulat un set complet de pachete cu starea legăturilor, el poate construi graful întregii subrețele, deoarece fiecare legătură este reprezentată. De fapt, fiecare legătură este reprezentată de două ori, o dată pentru fiecare direcție. Cele două valori pot fi mediate sau folosite separat.

Acum poate fi folosit local algoritmul lui Dijkstra, pentru a construi cea mai scurtă cale către toate destinațiile posibile. Rezultatul acestui algoritm poate fi trecut în tabelele de dirijare, iar apoi operațiile normale pot fi reluate.

Pentru o subrețea formată din *n* rutere, fiecare având *k* vecini, memoria necesară pentru a memora datele de intrare este proporțională cu *kn*. Pentru subrețele mari, aceasta poate fi o problemă. De asemenea și timpul de calcul poate fi important. Cu toate acestea, în multe situații, algoritmul bazat pe starea legăturilor funcționează bine.

Oricum, probleme ale hardware-ului sau software-ului pot influența negativ funcționarea acestui algoritm (la fel ca și a altora). De exemplu, dacă un ruter pretinde că are o linie pe care de fapt nu o are, sau uită despre o linie pe care o are, graful subrețelei va fi incorect. Dacă un ruter eșuează în retrimiterea pachetelor sau le alterează în timp ce le retrimite, vor apărea probleme. În fine, dacă un ruter rămâne fără memorie sau calculează greșit rutele, se vor petrece lucruri urâte. Pe măsură ce subrețeaua crește în dimensiune, ajungând la zeci sau sute de mii de noduri, probabilitatea ca un ruter să se defecteze devine neneglijabilă. Trucul care se poate folosi constă în încercarea de limitare a pagubelor atunci când inevitabilul s-a produs. Perlman (1988) tratează în detaliu aceste probleme, precum și soluțiile lor.

Dirijarea bazată pe starea legăturilor este larg folosită în rețelele actuale, astfel încât ar fi potrivite câteva cuvinte despre unele protocoale care o folosesc. Protocolul OSPF, care este extrem de folosit în Internet, utilizează un algoritm bazat pe starea legăturilor. Vom descrie protocolul OSPF în sect. 5.6.4.

Un alt protocol bazat pe starea legăturilor este **IS-IS** (**Intermediate System-Intermediate System,** Sistem Intermediar – Sistem Intermediar), care a fost proiectat pentru DECnet și mai apoi adoptat de ISO pentru a fi folosit cu protocolul neorientat pe conexiune de la nivelul rețea, CLNP. De atunci a fost modificat pentru a se descurca și cu alte protocoale, cel mai important fiind IP. IS-IS este folosit în coloane vertebrale ale Internet-ului (Internet backbones) (inclusiv în vechiul NSFNET) și în unele sisteme digitale celulare cum ar fi CDPD. Novell NetWare folosește o variantă simplificată IS-IS (NLSP) pentru a dirija pachetele IPX.

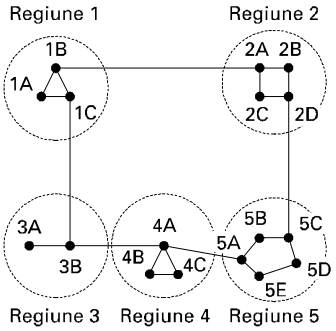
În principiu IS-IS distribuie o imagine a topologiei ruterelor, pe baza căreia se calculează calea cea mai scurtă. Fiecare ruter anunță, în informația de stare a legăturilor sale, ce adrese la nivelul rețea poate să acceseze direct. Aceste adrese pot fi IP, IPX, AppleTalk, sau orice alte adrese. IS-IS poate accepta chiar mai multe protocoale ale nivelului rețea în același timp.

Multe din inovațiile din proiectarea lui IS-IS au fost preluate de OSPF (OSPF a fost proiectat la mulți ani după IS-IS). Acestea includ o metodă auto-stabilizatoare a inundării, folosită la actualizarea stării legăturilor, conceptul de ruter dedicat într-o LAN, metoda de calcul și utilizare a căilor divizate și mai multe metrici. Ca o consecință, vor exista puține diferențe între IS-IS și OSPF. Cea mai importantă diferență este că IS-IS este codificat în așa fel, încât suportă ușor și chiar natural să transporte informațiile mai multor protocoale ale nivelului rețea, o caracteristică pe care OSPF nu o are. Acest avantaj este deosebit de important, în special în mediile mari, multiprotocol.

5.2.6 Dirijare ierarhică

Pe măsură ce rețelele cresc în dimensiune, tabelele de dirijare cresc proporțional. Pe lângă faptul că memoria ruterului este consumată de fiecare nouă creștere a tabelelor, pentru parcurgerea lor este necesar tot mai mult timp de calcul și se folosește tot mai mult din lățimea de bandă pentru a trimite rapoartele de stare despre ele. La un moment dat, rețeaua poate crește până la un punct în care nu mai este posibil ca fiecare ruter să dețină o intrare pentru fiecare alt ruter, astfel încât dirijarea trebuie făcută ierarhic, la fel ca în rețeaua telefonică.

Atunci când se folosește dirijarea ierarhică, ruterele sunt împărțite în ceea ce vom numi **regiuni** (**regions**), fiecare ruter știind toate detaliile necesare pentru a dirija pachete spre destinație în cadrul regiunii sale, dar neștiind nimic despre organizarea internă a celorlalte regiuni. Când rețele diferite sunt interconectate, este natural să privim fiecare rețea ca pe o regiune, pentru a elibera ruterele dintr-o rețea de sarcina de a cunoaște structura topologică a celorlalte.



(a)

Tabela completă pentru 1A

Dest.	Linie	Salturi
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Tabela ierarhică pentru 1A

Dest.	Linie	Salturi
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Fig. 5-15. Dirijare ierarhică.

Pentru rețele uriașe, o ierarhie pe două niveluri ar putea fi insuficientă, ar putea fi necesar să grupăm regiunile în asociații (clusters), asociațiile în zone, zonele în grupuri și așa mai departe până ce nu mai avem nume pentru aceste aglomerări. Ca un exemplu de ierarhie multinivel, să considerăm cum poate fi dirijat un pachet din Berkeley, California spre Malindi, Kenya. Ruterul din Berkeley cunoaște în detaliu topologia din California, așa că va trimite tot traficul pentru exteriorul statului către ruterul din Los Angeles. Ruterul din Los Angeles este capabil să dirijeze traficul spre alte rutere interne, dar va trimite tot traficul extern spre New York. Ruterul din New York este programat să dirijeze traficul către ruterul din țara de destinație care se ocupă de traficul extern, de exemplu în Nairobi. În final, pachetul va urma calea sa, coborând în arborele din Kenya până ce ajunge la Malindi.

Fig. 5-15 sugerează un exemplu cantitativ al dirijării într-o ierarhie pe două niveluri, având cinci regiuni. Tabela de dirijare completă a ruterului 1*A* are 17 intrări, așa cum se arată în fig. 5-15 (b). Atunci când dirijarea se face ierarhic, așa ca în fig. 5-15(c), există intrări pentru toate ruterele locale, la fel ca și până acum, însă toate celelalte regiuni au fost condensate într-un singur ruter, astfel încât tot traficul pentru regiunea 2 va merge pe linia 1*B*-2*A*, iar restul traficului la distanță va merge pe linia 1*C*-3*B*. Dirijarea ierarhică a redus dimensiunea tabelei de la 17 intrări la 7. Pe măsură ce raportul între numărul de regiuni și numărul de rutere dintr-o regiune crește, se economisește tot mai mult spațiu de memorie.

Din păcate acest câștig de spațiu nu este gratuit. Trebuie plătit un preț și acesta este concretizat în creșterea lungimii căilor. De exemplu, cea mai bună cale de la 1*A* la 5*C* este prin regiunea 2, însă în dirijarea ierarhică tot traficul către regiunea 5 este trimis spre regiunea 3, deoarece așa este mai bine pentru majoritatea destinațiilor din regiunea 5.

Dacă o rețea unică devine prea mare, o întrebare interesantă este: Câte niveluri trebuie să aibă ierarhia? De exemplu, să considerăm o subrețea cu 720 rutere. În absența oricărei ierarhii, tabela de

dirijare a fiecărui ruter trebuie să conțină 720 intrări. Dacă subrețeaua este partiționată în 24 regiuni cu 30 de rutere fiecare, fiecare ruter necesită 30 de intrări locale plus 23 de intrări pentru celelalte regiuni, deci un total de 53 intrări. Dacă se alege o ierarhie pe trei niveluri, cu opt asociații (clusters), fiecare având 9 regiuni a câte 10 rutere fiecare, fiecare ruter are nevoie de 10 intrări pentru ruterele locale, 8 intrări pentru celelalte regiuni din asociația sa și de 7 intrări pentru celelalte asociații, deci un total de 25 intrări. Kamoun și Kleinrock (1979) au descoperit că numărul optim de niveluri pentru o subrețea cu N rutere este $\ln N$, ceea ce necesită un total de $e \ln N$ intrări pentru fiecare ruter. De asemenea ei au arătat că creșterea efectivă a lungimii medii a căilor provocată de dirijarea ierarhică este suficient de mică pentru a fi acceptată.

5.2.7 Dirijarea prin difuzare

În unele aplicații, calculatoarele gazdă au nevoie să trimită mesaje către mai multe sau către toate celelalte calculatoare gazdă. De exemplu, un serviciu de distribuire a rapoartelor meteorologice, de actualizare a cursului acțiunilor sau transmisiuni radio în direct ar putea funcționa mai bine prin difuzarea datelor către toate mașinile, fiind la latitudinea celor interesate de date să le recepționeze. Trimiterea simultană a unui pachet către toate destinațiile se numește **difuzare** (**broadcast**); mai multe metode pentru realizarea ei au fost propuse.

O metodă de difuzare care nu are cerințe speciale pentru subrețea este ca sursa să trimită un pachet distinct către fiecare destinație. Metoda este nu numai consumatoare de lățime de bandă, dar ea cere ca sursa să dețină o listă completă a tuturor destinațiilor. S-ar putea ca în practică aceasta să fie singura metodă utilizabilă, însă ea este metoda cea mai puțin dorită.

Inundarea este un alt candidat evident. Deși inundarea este nepotrivită pentru comunicația obișnuită capăt la capăt, pentru difuzare ar trebui luată serios în considerare, mai ales dacă nici una dintre metodele ce vor fi descrise în continuare nu este aplicabilă. Problema folosirii inundării ca metodă de difuzare este aceeași cu problema ivită la folosirea sa ca algoritm de dirijare capăt la capăt: generează prea multe pachete și consumă lățime de bandă prea mare.

Al treilea algoritm este **dirijarea multidestinație** (**multidestination routing**). Dacă se folosește această metodă, fiecare pachet conține fie o listă a destinațiilor, fie o hartă de biți care indică destinațiile dorite. Atunci când un pachet ajunge la un ruter, ruterul verifică toate destinațiile pentru a determina setul liniilor de ieșire pe care trebuie trimis pachetul. (O linie de ieșire este selectată dacă este calea cea mai bună pentru cel puțin o destinație.) Ruterul generează o nouă copie a pachetului pentru fiecare linie de ieșire folosită și include în fiecare pachet doar acele destinații care folosesc linia respectivă. Efectul este partiționarea mulțimii destinațiilor între liniile de ieșire. După un număr suficient de salturi, fiecare pachet va conține o singură destinație și poate fi tratat ca un pachet normal. Dirijarea multidestinație este asemănătoare trimiterii separate a mai multor pachete către adresele destinație, cu excepția faptului că atunci când mai multe pachete trebuie să urmeze aceeași cale, unul dintre ele plătește tot drumul, iar celelalte călătoresc gratuit.

Al patrulea algoritm de difuzare utilizează explicit arborele de scufundare al ruterului care inițiază difuzarea sau orice alt arbore de acoperire util. Un **arbore de acoperire** (**spanning tree**) este un subset al subrețelei care include toate ruterele și nu conține bucle. Dacă fiecare ruter cunoaște care din liniile sale participă la arborele de acoperire, el poate copia un pachet de difuzare recepționat pe toate liniile de ieșire care fac parte din arborele de acoperire, cu excepția celei pe care a fost recepționat. Această metodă asigură o utilizare deosebit de eficientă a lățimii de bandă, generând numărul minim de pachete necesare pentru a rezolva problema. Singura problemă este că, pentru a fi aplica-

bilă, fiecare ruter trebuie să dețină cunoștințe despre un anume arbore de acoperire. Uneori această informație este disponibilă (de exemplu la dirijarea bazată pe starea legăturilor), iar alteori nu este disponibilă (de exemplu la dirijarea cu vectori distanță).

Ultimul nostru algoritm cu difuzare este o încercare de a aproxima comportamentul precedentului, chiar și atunci când ruterele nu știu absolut nimic despre arborele de acoperire. Ideea, numită **trimiterea pe calea inversă** (**reverse path forwarding**) este remarcabil de simplă odată ce a fost indicată. Când un pachet de difuzare ajunge la un ruter, acesta verifică dacă pachetul a sosit pe linia pe care se trimit de obicei pachete către sursa difuzării. Dacă este așa, este o șansă foarte mare ca însuși pachetul de difuzare să fi urmat cea mai bună cale, fiind astfel prima copie care ajunge la ruter. Aceasta fiind situația, ruterul trimite pachetul pe toate liniile de ieșire, cu excepția celei pe care a sosit. Dacă însă pachetul a sosit pe altă linie decât cea preferată pentru a ajunge la sursă, pachetul este distrus, fiind considerat un posibil duplicat.

Un exemplu de **trimitere pe calea inversă**, este prezentat în fig. 5-16. Partea (a) prezintă subrețeaua, partea (b) arată arborele de scufundare pentru ruterul *I* al subrețelei, iar partea (c) prezintă cum funcționează algoritmul căii inverse. La primul salt, *I* trimite pachete către *F*, *H*, *J* și *N*, așa cum se vede din al doilea nivel al arborelui. Fiecare din aceste pachete ajunge pe calea preferată către *I* (presupunând că ruta preferată face parte din arborele de scufundare), lucru care este indicat printr-un cerc în jurul literei. La saltul următor, se generează opt pachete, două de către fiecare ruter care a primit un pachet la primul salt. Așa cum se vede, toate aceste opt pachete ajung la rutere încă nevizitate și cinci dintre ele ajung pe linia preferată. Dintre cele șase pachete generate la al treilea salt, doar trei sosesc pe linia preferată (în *C, E* și *K*); celelalte sunt duplicate. După cinci salturi și 24 pachete, difuzarea se termină, spre deosebire de patru salturi și 14 pachete necesare dacă arborele de scufundare ar fi fost urmat integral.

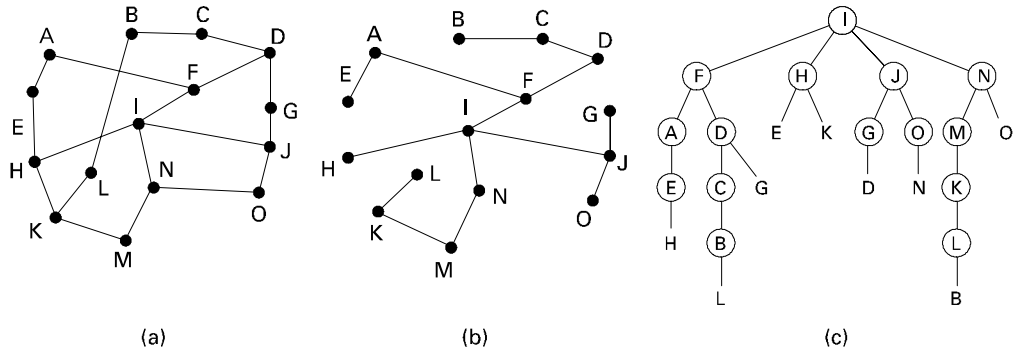


Fig. 5-16. Algoritmul trimiterii pe calea inversă. (a) O subrețea. (b) Un arbore de scufundare. (c) Arborele construit de algoritmul căii inverse.

Principalul avantaj al folosirii căii inverse este acela că este rezonabil de eficient și este ușor de implementat. El nu cere ruterelor să cunoască arborele de acoperire și nici nu are overhead-ul algoritmului de adresare multidestinație, care folosește o listă de destinații sau hartă de biți la fiecare difuzare. De asemenea, el nu necesită nici un mecanism special pentru a opri procesul, ca în cazul inundării (unde se folosește fie un contor al salturilor în fiecare pachet și o cunoaștere a priori a diametrului subrețelei, fie o listă de pachete deja văzute pentru fiecare sursă).

5.2.8 Dirijarea cu trimitere multiplă (multicast)

Unele aplicații necesită ca procese aflate la mari distanțe unele de altele să lucreze în grup, de exemplu, un grup de procese care implementează o bază de date distribuită. În aceste situații, este adesea necesar ca un proces să trimită un mesaj către toți ceilalți membri ai grupului. Dacă grupul este mic, el poate trimite fiecărui partener un mesaj capăt la capăt. Dacă grupul este mare, această strategie este costisitoare. Uneori se poate folosi difuzarea, dar folosirea difuzării pentru a anunța 1000 de mașini dintr-o rețea cu un milion de noduri este ineficientă, deoarece majoritatea receptorilor nu sunt interesați de mesaj (sau chiar mai rău, sunt foarte interesați, dar nu trebuie să vadă mesajul). De aceea, avem nevoie de o modalitate de a trimite mesaje spre grupuri bine definite, care conțin un număr mare de noduri, dar totuși redus față de dimensiunea întregii rețele.

Trimiterea unui mesaj către un astfel de grup se numește **multicasting**, iar algoritmul de dirijare asociat se numește **dirijare multicast** (**multicast routing**). În această secțiune, vom descrie o modalitate de a realiza dirijarea multicast. Pentru informații suplimentare, vezi (Chu et al., 2000; Costa et al. 2001; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

Dirijarea multicast necesită managementul grupului. Trebuie să existe modalități de a crea și distruge grupuri și de a permite proceselor să intre în grupuri sau să le părăsească. Cum se realizează aceste funcții nu este treaba algoritmului de dirijare. Important pentru algoritmul de dirijare este că atunci când un proces se atașează unui grup, el trebuie să informeze gazda sa despre aceasta. Este important ca ruterele să știe căror grupuri le aparțin calculatoarele gazdă asociate. Fie calculatoarele gazdă trebuie să anunțe ruterul asociat la producerea unei modificări în alcătuirea grupurilor, fie ruterul trebuie să interogheze periodic aceste calculatoare gazdă. În ambele cazuri, ruterul află din ce grupuri fac parte calculatoarele gazdă. Ruterele își informează vecinii, astfel că informația se propagă prin subrețea.

Pentru a realiza dirijarea multicast, fiecare ruter calculează arborele de acoperire care acoperă toate celelalte rutere din subrețea. De exemplu, în fig. 5-17(a) avem două grupuri, 1 și 2. Unele rutere sunt atașate la calculatoare gazdă care aparțin unuia sau ambelor grupuri, așa cum se indică în figură. Un arbore de acoperire pentru cel mai din stânga ruter este prezentat în fig. 5-17(b).

Atunci când un proces trimite un pachet multicast către un grup, primul ruter își examinează arborele de acoperire și îl retează, eliminând toate liniile care nu conduc către calculatoare gazdă, membre ale grupului. În exemplul nostru, fig. 5-17(c) arată arborele de acoperire retezat pentru grupul 1. Similar, fig. 5-17(d) arată arborele de acoperire retezat pentru grupul 2. Pachetele multicast sunt dirijate doar de-a lungul arborelui de acoperire corespunzător.

Sunt posibile mai multe moduri de retezare a arborelui de acoperire. Cel mai simplu se poate folosi dacă se utilizează dirijarea bazată pe starea legăturilor și fiecare ruter cunoaște întreaga topologie a subrețelei, inclusiv apartenența calculatoarelor gazdă la grupuri. Atunci arborele poate fi retezat pornind de la sfârșitul fiecărei căi, mergând spre rădăcină și eliminând toate ruterele care nu aparțin grupului respectiv.

În cazul dirijării folosind vectori distanță, poate fi aplicată o altă strategie de retezare a arborelui. Algoritmul de bază folosit este trimiterea pe calea inversă. Oricum, ori de câte ori un ruter fără nici un calculator gazdă interesat de un anume grup și fără nici o conexiune la alte rutere primește un mesaj multicast pentru acel grup, el va răspunde cu un mesaj PRUNE (retezare), anunțând expeditorul să nu îi mai trimită mesaje multicast pentru acel grup. Când un ruter care nu are printre calculatoarele gazdă nici un membru al vreunui grup primește astfel de mesaje pe toate liniile sale, el poate de asemenea să răspundă cu un mesaj PRUNE. În acest fel subrețeaua este retezată recursiv.

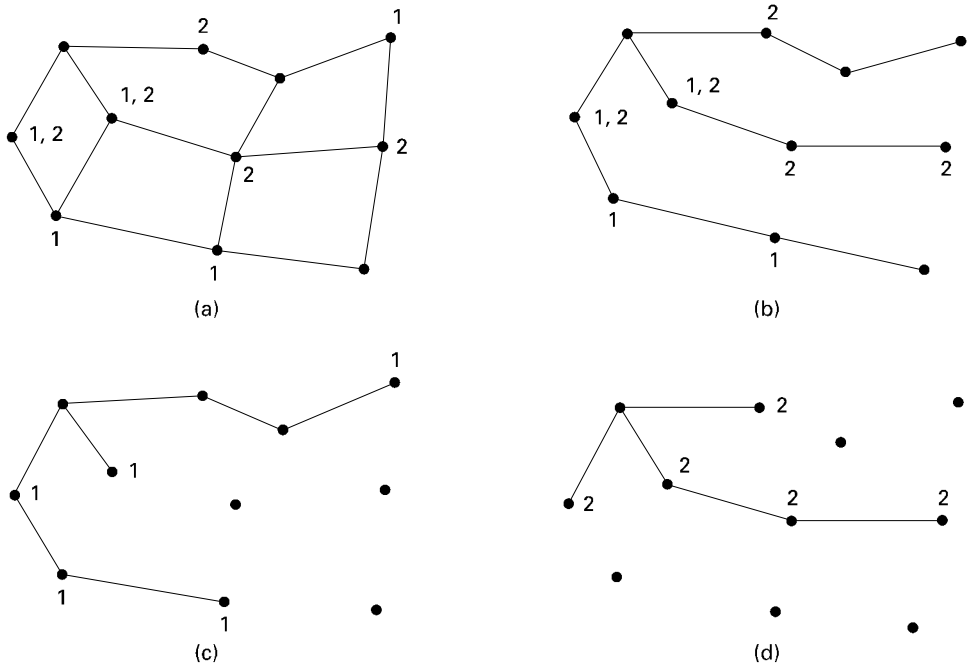


Fig. 5-17. (a) O rețea. (b) Un arbore de acoperire pentru cel mai din stânga ruter. (c) Un arbore multicast al grupului 1. (d) Un arborele multicast al grupului 2.

Un potențial dezavantaj al acestui algoritm este acela că se comportă deficitar în cazul rețelelor extinse. Să presupunem că o rețea are *n* grupuri, fiecare cu un număr mediu de *m* membri. Pentru fiecare grup, trebuie memorați *m* arbori de acoperire retezați, deci un total de *mn* arbori. Dacă există multe grupuri mari, atunci, pentru a memora toți arborii, este necesar un spațiu de memorie considerabil.

O alternativă de proiectare folosește **arbori de bază** (**core-base trees**) (Ballardie ș.a., 1993). Aici se calculează un singur arbore de acoperire pentru fiecare grup, având rădăcina (core - inima, nucleu) lângă mijlocul grupului. Pentru a trimite un mesaj multicast, un calculator gazdă trimite mesajul către rădăcină, care apoi îl trimite de-a lungul arborelui de acoperire. Deși acest arbore nu va fi optim pentru toate sursele, reducerea costului memorării de la *m* arbori la unul singur pe grup reprezintă o economie semnificativă.

5.2.9 Dirijarea pentru calculatoare gazdă mobile

Milioane de oameni dețin astăzi calculatoare portabile și, de obicei, doresc să-și citească poșta electronică și să-și acceseze sistemele de fișiere uzuale din orice punct al lumii s-ar afla. Aceste calculatoare mobile introduc o nouă complicație: pentru a dirija un pachet către un calculator mobil, rețeaua trebuie mai întâi să-l localizeze. Problema integrării calculatoarelor mobile într-o rețea este foarte recentă, dar în această secțiune vom sugera unele abordări și vom da o posibilă soluție.

Modelul lumii folosit de obicei de proiectanții de rețele este cel prezentat în fig. 5-18. Aici avem o rețea WAN formată din rutere și calculatoare gazdă. La WAN sunt conectate LAN-uri, MAN-uri și celule de comunicație fără fir de tipul celor descrise în Cap. 2.

Calculatoarele gazdă care nu se mișcă niciodată se numesc **staționare**. Ele sunt conectate la rețea prin fire de cupru sau fibre optice. Prin contrast, putem distinge alte două categorii de calculatoare gazdă. Calculatoarele gazdă **migratoare** sunt de fapt calculatoare gazdă staționare, dar care, din când în când, se mută dintr-un loc fixat în altul și care folosesc rețeaua doar atunci când sunt conectate fizic. Calculatoarele gazdă **călătoare** efectuează calcule în timp ce se deplasează și doresc să mențină legătura în timpul deplasării. Vom folosi termenul **gazdă mobilă** pentru a desemna fiecare dintre ultimele două categorii, adică toate calculatoarele gazdă care sunt departe de casă și doresc totuși să fie conectate.

Se presupune că toate calculatoarele gazdă au o **locație de domiciliu** permanentă (home location), care nu se modifică niciodată. Calculatoarele gazdă au de asemenea o adresă personală permanentă, care poate fi folosită pentru a determina locația de domiciliu, într-o manieră similară celei în care numărul de telefon 1-212-5551212 indică Statele Unite (codul de țară 1) și Manhattan (212). Scopul dirijării în sistemele cu calculatoare gazdă mobile este de a face posibilă trimiterea pachetelor spre gazde mobile folosind adresa lor personală permanentă și să se asigure o trimitere eficientă a pachetelor spre ele, oriunde ar fi acestea. Problema este, bineînțeles, modul de localizare a lor.

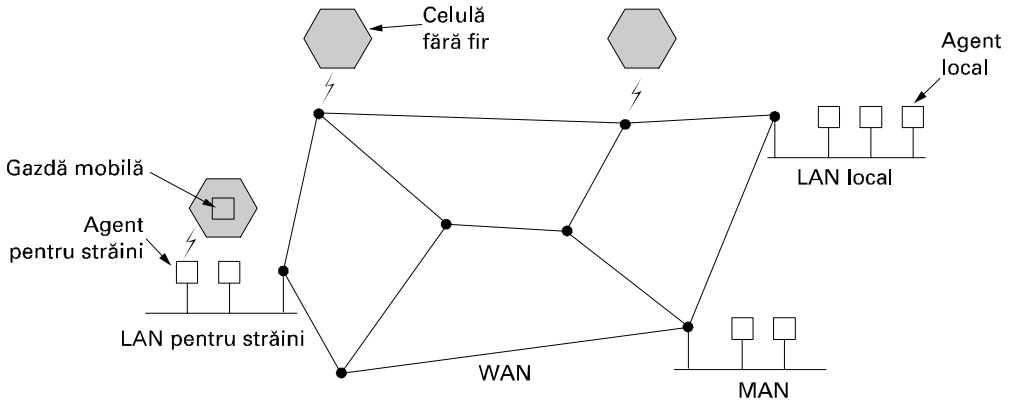


Fig. 5-18. O rețea WAN la care sunt conectate rețele LAN, MAN și celule de comunicație fără fir.

În modelul din fig. 5-18, lumea este divizată (geografic) în unități de dimensiune redusă. Să denumim aceste unități domenii, unde un domeniu este de obicei un LAN sau o celulă de comunicație fără fir. Fiecare domeniu are unul sau mai mulți **agenți pentru străini (foreign agent)**, procese ce țin evidența tuturor gazdelor mobile care vizitează domeniul. În plus, fiecare domeniu are un **agent local (local agent)** care ține evidența calculatoarelor gazdă cu domiciliul în domeniul respectiv, dar care momentan vizitează alte domenii.

Când un nou calculator gazdă pătrunde într-un domeniu, fie prin conectarea la acesta (atașarea fizică la LAN), fie prin plimbarea printr-o celulă, trebuie să se înregistreze la agentul pentru străini din domeniul respectiv. Procedura de înregistrare se desfășoară de obicei astfel:

1. Periodic, fiecare agent pentru străini difuzează un pachet anunțându-și existența și adresa. Un utilizator mobil nou sosit trebuie să aștepte unul dintre aceste mesaje, însă dacă nici

unul nu sosește într-un interval rezonabil de timp, calculatorul mobil poate difuza un pachet care spune: „Este vreun agent pentru străini prin zonă?”

2. Calculatorul mobil se înregistrează la agentul pentru străini precizând adresa sa permanentă, adresa actuală a nivelului legătură de date, precum și unele informații de securitate.
3. Agentul pentru străini contactează apoi agentul local al domeniului din care provine utilizatorul mobil și îi spune: „Unul dintre calculatoarele tale gazdă se află chiar aici.” Mesajul agentului pentru străini către agentul local conține adresa de rețea a agentului pentru străini. El mai conține de asemenea și informația de securitate, pentru a convinge agentul local că gazda mobilă este într-adevăr acolo.
4. Agentul local examinează informația de securitate, care conține și o ștampilă de timp, pentru a dovedi că a fost generată în ultimele câteva secunde. Dacă este mulțumit, atunci anunță agentul pentru străini că poate trece la acțiune.
5. Când agentul pentru străini primește confirmarea de la agentul local, el creează o nouă intrare în tabela sa și anunță utilizatorul mobil că a fost înregistrat.

Ideal, atunci când un calculator gazdă părăsește un domeniu, acest lucru trebuie anunțat, pentru a fi scos din evidență, însă mulți utilizatori pur și simplu închid calculatorul când termină.

Când se trimite un pachet către o gazdă mobilă, el este dirijat către domiciliul calculatorului gazdă, deoarece adresa sugerează că așa trebuie făcut, după cum se ilustrează și în pasul 1 din fig. 5-19. Aici emițătorul, aflat în orașul de nord-vest Seattle, dorește să trimită un pachet către un calculator gazdă aflat în New York, traversând Statele Unite. Pachetele trimise către calculatorul gazdă mobil în LAN-ul său de domiciliu, în New York, sunt interceptate de agentul local de acolo. Apoi agentul local caută noua locație (temporară) a gazdei mobile și află adresa agentului pentru străini care se ocupă de gazdele mobile, în Los Angeles.

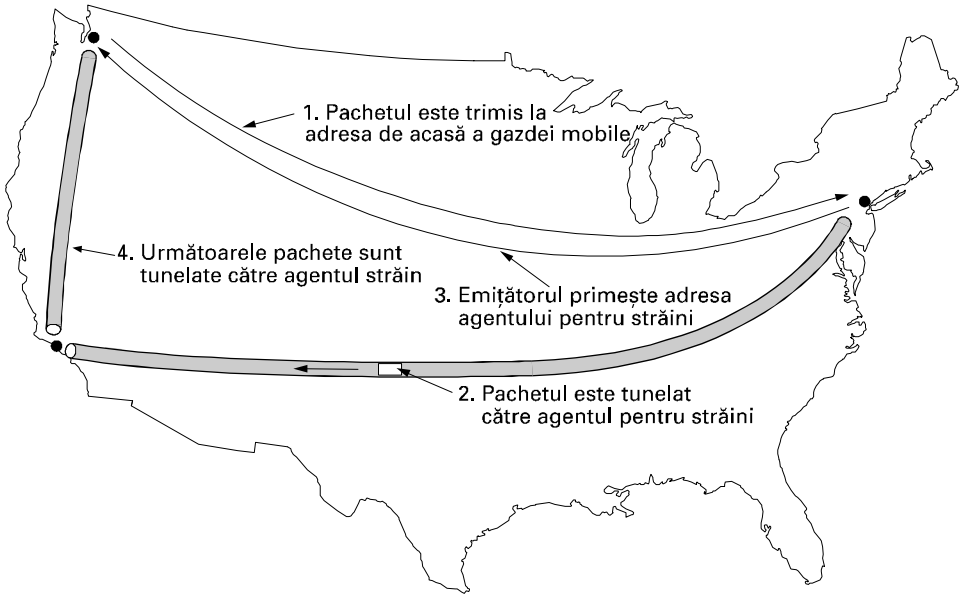

1. Pachetul este trimis la adresa de acasă a gazdei mobile
4. Următoarele pachete sunt tunelate către agentul străin
3. Emițătorul primește adresa agentului pentru străini
2. Pachetul este tunelat către agentul pentru străini

Fig. 5-19. Dirijarea pachetelor pentru gazde mobile.

Agentul local face două lucruri. În primul rând, încapsulează pachetul în câmpul de informație utilă (payload) al unui pachet de trimis, pe care îl expediază apoi către agentul pentru străini (pasul 2

din fig. 5-19). Acest mecanism se numește tunelare; îl vom studia în detaliu puțin mai târziu. După ce recepționează pachetul încapsulat, agentul pentru străini extrage pachetul inițial din câmpul payload și îl trimite către calculatorul gazdă mobil drept cadru al nivelului legătură de date.

În al doilea rând, agentul local anunță expeditorul mesajului ca de acum încolo să trimită pachetele adresate gazdei mobile încapsulându-le în câmpul de informație utilă (payload) al unor pachete trimise explicit agentului pentru străini, în loc să le trimită la adresa de domiciliu a calculatorului gazdă mobil (pasul 3). Pachetele următoare vor putea fi dirijate direct către utilizator, prin intermediul agentului pentru străini (pasul 4), evitând trecerea prin locația de domiciliu.

Diversele scheme propuse diferă prin mai multe aspecte. În primul rând, cât din acest protocol este realizat de rutere și cât de calculatoarele gazdă și, în această ultimă situație, pe care nivel de pe calculatorul gazdă. În al doilea rând, în unele scheme există rutere de-a lungul traseului care înregistrează adresele modificate, astfel încât să fie posibilă interceptarea și redirectarea traficului chiar înainte ca acesta să ajungă la locația de domiciliu. În al treilea rând, în unele variante fiecare vizitator primește o adresă temporară unică; în altele adresa temporară se referă la un agent care se ocupă de toți vizitatorii.

În al patrulea rând, variantele propuse diferă prin modul de rezolvare a situației în care pachetele adresate unei destinații trebuie livrate în altă parte. O variantă este schimbarea adresei destinație și retransmiterea pachetului modificat. Ca o alternativă, întregul pachet, adresa de domiciliu și toate celelalte pot fi încapsulate în câmpul de informație utilă (payload) al altui pachet care este trimis spre adresa temporară. În fine, schemele diferă prin aspectele legate de securitate. În general, când un ruter sau calculator gazdă primește un mesaj de forma „Începând din acest moment, vă rog să trimiteți toate mesajele de poștă electronică ale lui Stephany către mine," el ar putea avea dubii în legătură cu partenerul de dialog și dacă este o idee bună. Diferite protocoale pentru calculatoare gazdă mobile sunt discutate și comparate în (Hac și Guo, 2000; Perkins, 1998a; Snoeren și Balakrishan, 2000; Solomon, 1998; și Wang și Chen, 2001).

5.2.10 Dirijarea în rețele AD HOC

Am văzut cum se face dirijarea atunci când calculatoarele gazdă sunt mobile, dar ruterele sunt fixe. Un caz aparte apare atunci când însăși ruterele sunt mobile. Printre situațiile posibile sunt:

1. Vehicule militare pe un câmp de luptă atunci când nu există o infrastructură.
2. O flotă aflată în larg.
3. Echipe de intervenție la un cutremur ce a distrus infrastructura.
4. O adunare de persoane cu calculatoare portabile într-o zonă fără 802.11.

În toate aceste cazuri, și altele, fiecare nod este compus dintr-un ruter și o gazdă, de obicei pe același calculator. Rețelele de noduri ce întâmplător se află aproape unul de celălalt sunt numite **rețele ad hoc** (**ad hoc network**) sau **MANET** (**Mobile Ad hoc NETworks**, rețele mobile ad hoc). Să le studiem pe scurt. Mai multe informații pot fi găsite în (Perkins, 2001).

Diferența între rețelele ad hoc și rețelele cablate (eng.: wired) este că toate regulile despre topologii fixe, vecini ficși și cunoscuți, relații fixe stabilite între adresa IP și locație, și altele, sunt complet suprimate. Cât ai clipi ruterele pot să apară și să dispară sau să apară în alte locuri. În cazul rețelelor cablate, dacă un ruter are o cale validă către o destinație, aceasta continuă sa fie validă un timp nedefinit (suportând o defecțiune în altă parte a sistemului). Cu o rețea ad hoc, topologia se poate schimba mereu, așa că oportunitatea și chiar validitatea cailor se poate schimba spontan, fără averti-

zare. Nu mai trebuie spus că în aceste circumstanțe dirijarea în rețelele ad hoc diferă foarte mult față de dirijarea în echivalentul lor fix.

Au fost propuși diferiți algoritmi de dirijare pentru rețelele ad hoc. Unul dintre cei mai interesanți este algoritmul de dirijare **AODV** (**Ad hoc On-demand Distance Vector**, vectori distanță ad hoc la cerere) (Perkins și Royer, 1999). Este o rudă îndepărtată a algoritmului cu vectori distanță Bellman-Ford dar adaptat pentru un mediu mobil și care ia în considerare limitele lărgimii de bandă și durata scurtă de funcționare a unei baterii în acest mediu. O altă caracteristică neobișnuită este faptul că acesta este un algoritm la cerere, adică determină o cale către o anumită destinație doar atunci când cineva dorește să trimită un pachet către acea destinație. Să vedem acum ce înseamnă asta.

Descoperirea rutei

La orice moment de timp, o rețea ad hoc poate fi descrisă de un graf de noduri (rutere + gazde). Două noduri sunt conectate (de exemplu, există un arc ce le unește în graf) dacă pot comunica direct folosind radioul. Cum unul dintre ele poate avea un transmițător mai puternic decât celălalt, este posibil ca *A* să fie conectat cu *B* dar *B* să nu fie conectat cu *A*. Totuși, pentru a simplifica, presupunem că toate conexiunile sunt simetrice. De asemenea ar trebui observat că simplul fapt că fiecare din cele două noduri este în raza radio a celuilalt nu înseamnă că ele sunt conectate. Pot exista clădiri, dealuri sau alte obstacole care să blocheze comunicația.

Pentru a descrie algoritmul, considerăm rețeaua ad hoc din fig. 5-20, în care un proces al nodului *A* dorește să transmită un pachet nodului *I*. Algoritmul AODV menține o tabelă în fiecare nod, în care cheia este destinația, și care dă informații despre acea destinație, inclusiv cărui vecin trebuie trimise pachetele pentru a ajunge la destinație. Să presupunem că *A* se uită în tabelă și nu găsește nici o intrare pentru *I*. Trebuie deci să descopere o rută până la *I*. Această proprietate de descoperire a rutelor doar atunci când sunt necesare este cea care face din acest algoritm un algoritm „la cerere”.

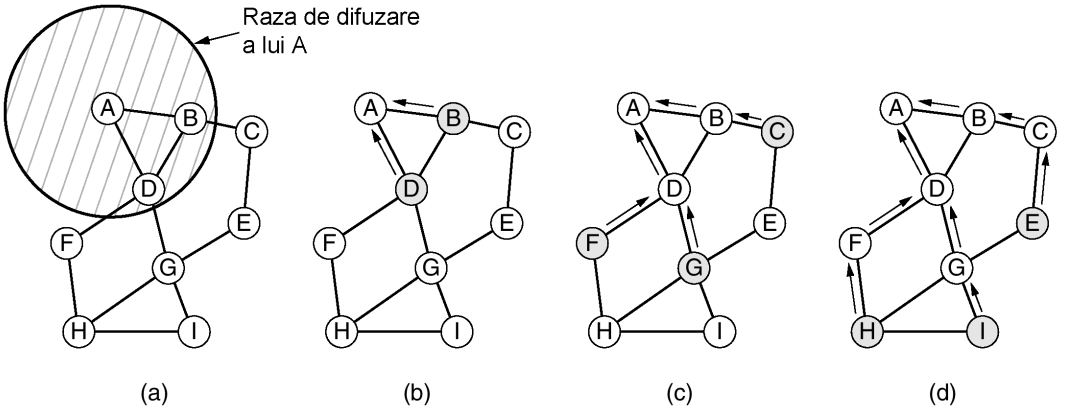


Fig. 5-20. (a) Raza de difuzare a lui *A*. (b) După ce *B* și *D* au primit difuzarea lui *A*. (c) După ce *C, F* și *G* au primit difuzarea lui *A*. (d) După ce *E, H* și *I* au primit difuzarea lui *A*. Nodurile hașurate sunt noii destinatari. Săgețile arată calea inversă posibilă.

Pentru a îl localiza pe *I*, *A* construiește un pachet special ROUTE REQUEST și îl difuzează. Pachetul ajunge la *B* și la *D,* așa cum este ilustrat în fig. 5-20(a). De fapt, motivul pentru care *B* și *D* sunt conectate cu *A* în graf este acela că pot comunica cu *A*. *F*, de exemplu, nu este prezentat ca fiind legat printr-un arc cu *A* deoarece nu poate recepționa semnalul radio emis de *A*.

Formatul pachetului ROUTE REQUEST este prezentat în fig. 5-21. El conține adresele sursă și destinație, bineînțeles adrese IP, care specifică cine caută pe cine. Conține de asemenea și un *ID Cerere* (Request ID), care este un contor local ținut de fiecare nod și incrementat la fiecare difuzare a unui pachet ROUTE REQUEST. Împreună, câmpurile *adresă* Sursă și *ID Cerere* identifică unic pachetul ROUTE REQUEST pentru a permite nodurilor să înlăture orice duplicat pe care-l primesc.

Adresa sursei	ID cerere	Adresa destinației	Numărul de secvență al sursei	Numărul de secvență al destinației	Contorul de salturi

Fig. 5-21. Formatul unui pachet ROUTE REQUEST

În plus, pe lângă contorul *ID Cerere*, fiecare nod menține un al doilea contor de secvență incrementat de fiecare dată când este trimis un pachet ROUTE REQUEST (sau un răspuns la un ROUTE REQUEST). Acesta funcționează asemănător unui ceas și este folosit pentru a deosebi rutele noi de cele vechi. Cel de-al patrulea câmp din fig. 5-21. este contorul de secvențe al lui *A*; al cincilea câmp este cea mai recentă valoare a contorului de secvențe a lui *I* întâlnită de *A* (0 dacă nu a întâlnit-o niciodată). Rolul acestor câmpuri se va clarifica în scurt timp. Ultimul câmp, *contorul de salturi* (Hop count), va memora câte salturi a făcut pachetul. Acesta este inițializat cu 0.

Când un pachet ROUTE REQUEST ajunge la un nod (în acest caz *B* și *D*), este prelucrat astfel:

1. Perechea (Adresă sursă, ID Cerere) este căutată într-o tabelă locală cu istoria cererilor pentru a vedea dacă această cerere a mai fost întâlnită și procesată. Dacă este un duplicat, atunci este înlăturat și procesarea se oprește. Dacă nu este un duplicat, atunci perechea este introdusă tabelă pentru ca viitoarele duplicate să poată fi rejectate, și procesarea continuă.
2. Receptorul caută destinația în propria tabela de dirijare. Dacă a aflat o nouă cale către destinație, atunci sursei îi este trimis un pachet ROUTE REPLY, spunându-i cum se ajunge la destinație. Nouă înseamnă că *numărul de secvență al destinației* memorat în tabela de dirijare este mai mare sau egal cu numărul de secvență al destinației din pachetul ROUTE REQUEST. Dacă este mai mic, calea memorată este mai veche decât calea anterioară pe care sursa o avea pentru destinație, așa că se execută pasul 3.
3. Deoarece receptorul nu știe o rută nouă către destinație, incrementează câmpul *contor de salturi* și redifuzează pachetul ROUTE REQUEST. De asemenea extrage datele din pachet și le memorează ca o nouă intrare în tabela de căi inverse. Această informație va fi folosită la construcția caii inverse pentru ca răspunsul să se poată întoarce mai târziu la sursă. Săgețile din fig. 5-20 sunt folosite pentru construirea căii inverse. De asemenea este pornit un cronometru asociat intrării corespunzătoare căii inverse nou create. Dacă expiră, intrarea este ștearsă.

Nici B și nici D nu știu unde este I, așa că fiecare creează o intrare pentru calea inversă indicând către A, așa cum arată săgețile din fig. 5-20, și difuzează pachetul cu contorul de salturi setat pe 1. Difuzarea de la B ajunge la C și D. C creează o intrare pentru el în tabela de căi inverse și-l redifuzează. În contrast, *D* îl rejectează ca pe un duplicat. Similar, difuzarea făcută de *D* este rejectată de *B*. Totuși, difuzarea făcută de *D* este acceptată de *F* și *G* și memorată, așa cum este prezentat în fig. 5-20(c). După ce *E*, *H* și *I* primesc difuzarea, pachetul ROUTE REQUEST ajunge în final la o destinație care știe unde se află *I*, de fapt, chiar *I*, așa cum este ilustrat în fig. 5-20(d). Observați că deși am prezentat difuzările în trei pași discreți, difuzările de la noduri diferite nu sunt coordonate în nici un fel.

Ca răspuns la cererea venită, *I* construiește un pachet ROUTE REPLY, prezentat în fig. 5-22. *Adresa sursă, Adresa destinație* și *Contorul de salturi* sunt copiate din cererea venită, dar *Numărul de secvență al destinației e*ste luat din contorul său din memorie. Contorul de salturi este setat la 0. Câmpul *Durată de viață* controlează perioada în care ruta este validă. Acest pachet este trimis doar către nodul de la care a venit pachetul ROUTE REQUEST, în acest caz, *G*. Apoi urmează calea inversă către *D* și în final către *A*. La fiecare nod, *Contorul de salturi* este incrementat astfel încât nodul să poată vedea cât de departe de destinație (*I*) este.

Adresa sursei	Adresa destinației	Numărul de secvență al destinației	Contorul de salturi	Durata de viață

Fig. 5-22. Formatul pachetului ROUTE REPLY

Pachetul este analizat la fiecare nod intermediar de pe drumul înapoi. El este introdus în tabela locală de dirijare ca rută către *I* dacă una sau mai multe din cele trei condiții de mai jos este îndeplinită:

1. Nu se cunoaște nici o rută către *I*.
2. Numărul de secvență pentru *I* din pachetul ROUTE REPLY este mai mare decât valoarea din tabela de dirijare.
3. Numerele de secvență sunt egale, dar noua rută este mai scurtă.

În acest mod, toate nodurile de pe calea inversă află și ruta către *I*, ca o consecință a descoperirii rutei de către *A.* Nodurile care au primit pachetul ROUTE REQUEST original, dar nu sunt pe calea inversă (*B, C, E, F,* și *H* în acest exemplu) vor înlătura intrarea din tabela de căi inverse la expirarea timpului asociat.

Într-o rețea mare, algoritmul generează multe difuzări, chiar pentru destinații aflate foarte aproape. Numărul de difuzări poate fi redus după cum urmează. *Durata de viață* a pachetului IP este inițializată de emițător cu valoarea diametrului rețelei și decrementată la fiecare salt. Dacă ajunge la valoarea 0, pachetul este înlăturat în loc să fie difuzat.

Procesul descoperirii este modificat după cum urmează. Pentru a localiza o destinație, emițătorul difuzează un pachet ROUTE REQUEST cu *Durata de viață* setată la 1. Dacă nu primește nici un răspuns într-un timp rezonabil, este trimis un alt pachet, de această dată cu *Durata de viață* setată la 2. La încercările următoare se utilizează 3, 4, 5 etc. În acest fel se încearcă întâi local, apoi în cercuri din ce în ce mai largi.

Întreținerea rutei

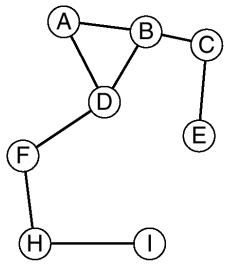
Deoarece nodurile se pot muta sau pot fi oprite, topologia se poate schimba spontan. De exemplu, în fig. 5-20, dacă *G* este oprit, *A* nu își va da seama că ruta pe care o folosea către *I* (*ADGI*) nu mai este validă. Algoritmul trebuie să poată rezolva această situație. Periodic, fiecare nod difuzează un mesaj *Hello.* Fiecare vecin ar trebui să răspundă. Dacă nu este primit nici un răspuns, atunci emițătorul știe că respectivul vecin s-a mutat din raza sa și nu mai este conectat cu el. Similar, dacă încearcă să-i trimită un pachet unui vecin ce nu răspunde, află că vecinul nu mai este disponibil.

Această informație este folosită pentru a elimina rutele care nu mai funcționează. Pentru fiecare destinație posibilă, fiecare nod, *N*, memorează vecinii care au trimis un pachet către acea destinație în ultimele ΔT secunde. Aceștia sunt numiți **vecinii activi** (**active neighbors**) ai lui *N* . *N* realizează acest lucru prin intermediul unei tabele de dirijare în care cheia este destinația și care conține nodul de ieșire ce trebuie utilizat pentru a ajunge la destinație, contorul de salturi până la destinație, cel mai recent

număr de secvență al destinației și lista vecinilor activi pentru acea destinație. O tabelă de dirijare posibilă pentru nodul *D* pentru topologia din exemplul nostru este prezentată în fig. 5-23(a).

Dest.	Următorul salt	Distanța	Vecini activi	Alte câmpuri
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Fig. 5-23. (a) Tabela de dirijare a lui *D* înaintea opririi lui *G*. (b) Graful după ce *G* s-a oprit.

Când oricare dintre vecinii lui *N* devine inaccesibil, el verifică tabela de dirijare pentru a vedea ce destinații au rute ce trec prin vecinul dispărut. Pentru fiecare din aceste rute, vecinii activi sunt informați că ruta ce trece prin *N* este acum inutilizabilă și trebuie ștearsă din tabela lor de dirijare. Vecinii activi transmit mai departe informația vecinilor lor activi, și așa mai departe, recursiv, până ce toate rutele ce depindeau de nodul devenit inaccesibil sunt șterse din toate tabelele de dirijare.

Ca exemplu de întreținere a rutelor, să considerăm exemplul anterior, dar cu *G* oprit brusc. Topologia schimbată este ilustrată în fig. 5-23(b). Când *D* descoperă că *G* nu mai este în rețea, se uită în tabela de dirijare și observă că *G* era folosit pentru rutele către *E, G,* și *I.* Mulțimea vecinilor activi pentru aceste destinații este {*A, B*}. Cu alte cuvinte, *A* și *B* depind de *G* pentru câteva dintre rutele lor, așa că ele trebuie informate că aceste rute nu mai sunt valide. *D* le informează prin trimiterea de pachete care le determină să-și actualizeze tabelele de dirijare. De asemenea, *D* șterge intrările pentru *E, G,* și *I* din tabela sa de dirijare.

Poate nu a fost evident din descrierea noastră, dar o diferență critică între AODV și Bellman-Ford este că nodurile nu difuzează periodic întreaga lor tabelă de dirijare. Această diferență economisește atât lățimea de bandă, cât și durata de viață a bateriei.

De asemenea, AODV este capabil de dirijare prin difuzare și de dirijare cu trimitere multiplă. Pentru detalii, consultați (Perkins și Royer, 2001). Dirijarea ad hoc este un domeniu de cercetare fierbinte. S-a publicat mult despre acest subiect. Câteva dintre lucrări sunt (Chen et. al, 2002; Hu și Johnson, 2001; Li et. al, 2001; Raju și Garcia-Luna-Aceves, 2001; Ramanathan și Redi, 2002; Royer și Toh, 1999; Spohn și Garcia-Luna-Aceves, 2001; Tseng et. al, 2001; și Zadeh et. al, 2002).

5.2.11 Căutarea nodurilor în rețele punct la punct

Un fenomen relativ nou este reprezentat de rețelele punct la punct, în care un număr mare de persoane, de obicei cu conexiuni permanente la Internet, sunt în contact pentru a partaja resurse. Prima aplicație larg răspândită a tehnologiei punct la punct a fost un delict de anvergură: 50 de milioane de utilizatori ai Napster-ului făceau schimb de cântece cu copyright fără permisiunea proprietarilor până când utilizarea Napster-ului a fost interzisă de tribunal în vâltoarea unor mari controverse. Totuși, tehnologia punct la punct are multe utilizări interesante și legale. De asemenea, are

ceva asemănător cu problema dirijării, deși nu este chiar la fel cu cele studiate până acum. Cu toate acestea, merită să-i acordăm puțină atenție.

Ceea ce face ca sistemele punct la punct să fie interesante este faptul că ele sunt în întregime distribuite. Toate nodurile sunt simetrice și nu există un control central sau o ierarhie. Într-un sistem punct la punct tipic fiecare utilizator are o informație ce poate fi de interes pentru alți utilizatori. Această informație poate fi software gratuit, muzică (din domeniul public), fotografii și așa mai departe. Dacă numărul de utilizatori este mare, aceștia nu se știu între ei și nu știu unde să găsească ceea ce caută. O soluție este o bază de date centrală, dar așa ceva ar putea fi irealizabil din diverse motive (de exemplu, nimeni nu dorește să o găzduiască și s-o administreze). Astfel, problema se reduce la felul în care un utilizator găsește un nod ce conține ceea ce caută el, în absența unei baze de date centralizate sau a unui index centralizat.

Să presupunem că un utilizator are una sau mai multe unități de date cum ar fi cântece, fotografii, programe, fișiere ș.a.m.d, pe care ar dori să le citească alți utilizatori. Fiecare unitate are ca nume un șir de caractere ASCII. Un posibil utilizator cunoaște numai șirul de caractere ASCII și vrea să afle dacă una sau mai multe persoane au copii, și în caz că au, care sunt adresele IP ale acestora.

De exemplu, să considerăm o bază de date genealogică distribuită. Fiecare specialist în genealogie are o serie de înregistrări on-line despre strămoșii și rudele sale, posibil și cu fotografii, clipuri audio sau chiar video ale persoanei. Mai multe persoane pot avea același străbunic, astfel că un strămoș poate avea înregistrări în mai multe noduri. Numele înregistrării este numele persoanei într-o formă canonică. La un moment dat, un specialist în genealogie descoperă testamentul străbunicului său într-o arhivă, în care străbunicul lasă moștenire nepotului ceasul de buzunar de aur. Genealogistul știe numele nepotului și dorește să afle dacă alt genealogist are o înregistrare pentru el. Cum aflăm, fără o bază de date centrală, cine are, dacă are cineva, aceste înregistrări?

Pentru a rezolva această problemă au fost propuși diverși algoritmi. Cel pe care îl vom studia este Chord (Dabek ș.a.,2001; și Stoica ș.a.,2001). O explicație simplificată a felului în care funcționează este următoarea. Sistemul Chord constă din n utilizatori participanți, fiecare dintre ei având câteva înregistrări memorate și fiecare dintre ei fiind pregătit să memoreze fragmente din index pentru uzul celorlalți utilizatori. Fiecare nod utilizator are o adresă IP care poate fi convertită într-un număr pe m biți, folosind o funcție de dispersie, *hash*. Pentru *hash* Chord folosește SHA-1. SHA-1 este folosit în criptografie; îl vom analiza în Cap. 8. Pentru moment, este doar o funcție care primește ca argument un șir de caractere de lungime variabilă și generează un număr aleator pe 160 de biți. Deci, putem converti orice adresă IP într-un număr pe 160 de biți numit **identificatorul nodului** (**node identifier**).

Conceptual, toți cei 2^{160} identificatori de noduri sunt așezați în ordine crescătoare într-un mare cerc. Câțiva corespund nodurilor participante, dar majoritatea nu. În fig. 5-24(a) prezentăm un cerc de identificatori de noduri pentru $m = 5$ (pentru moment ignorați arcele din centru). În acest exemplu, nodurile cu identificatorii 1, 4, 7, 12, 15, 20 și 27 corespund nodurilor reale și sunt hașurate; restul nodurilor nu există.

Să definim acum funcția *successor(k), ca* identificator al primului nod real ce urmează după k pe cerc, în sensul acelor de ceasornic. De exemplu, successor(6) = 7, successor(8) = 12 și successor(22) = 27.

Numele înregistrărilor (nume de cântece, numele strămoșilor etc.) sunt de asemenea convertite cu funcția *hash* (adică SHA-1) pentru a genera un număr pe 160 de biți, numit **cheie (key)**. Astfel, pentru a converti un *nume* (numele ASCII al înregistrării) la cheia sa, folosim *key = hash (name)*. Acest calcul este doar un apel local al procedurii *hash*. Dacă o persoană deținând o înregistrare genealogică pentru *nume* dorește să o facă publică, atunci construiește mai întâi un tuplu (*nume, adresă*

IP) și apoi îl roagă pe *successor(hash(nume))* să-l memoreze. Dacă există mai multe înregistrări (în noduri diferite) pentru același nume, toate tuplurile lor vor fi memorate în același nod. În acest mod, index-ul este distribuit aleator pe noduri. Pentru toleranța la defecte, pentru memorarea fiecărui tuplu în *p* noduri pot fi folosite *p* funcții de dispersie diferite, dar nu ne vom referi aici la acest aspect.

Dacă ulterior un utilizator caută *nume*, îi aplică funcția de dispersie pentru a afla cheia și apoi folosește *successor(cheie)* pentru a afla adresa IP a nodului ce memorează tuplurile respective din index. Primul pas este ușor; dar al doilea nu. Pentru a face posibilă găsirea adresei IP corespunzătoare unei anumite chei, fiecare nod trebuie să mențină structuri de date administrative. Una dintre acestea este adresa IP a nodului succesor, de-a lungul cercului de identificatori de noduri. De exemplu, în fig. 5-24, succesorul nodului 4 este 7 și succesorul nodului 7 este 12.

Căutarea se poate desfășura după cum urmează. Nodul care face cererea trimite succesorului său un pachet conținând adresa sa IP și cheia pe care o caută. Pachetul se propagă prin inel până ce localizează succesorul identificatorului căutat. Nodul verifică dacă deține vreo informație cu cheia respectivă și, dacă este așa, o returnează direct nodului care a făcut cererea, a cărui adresă IP o are.

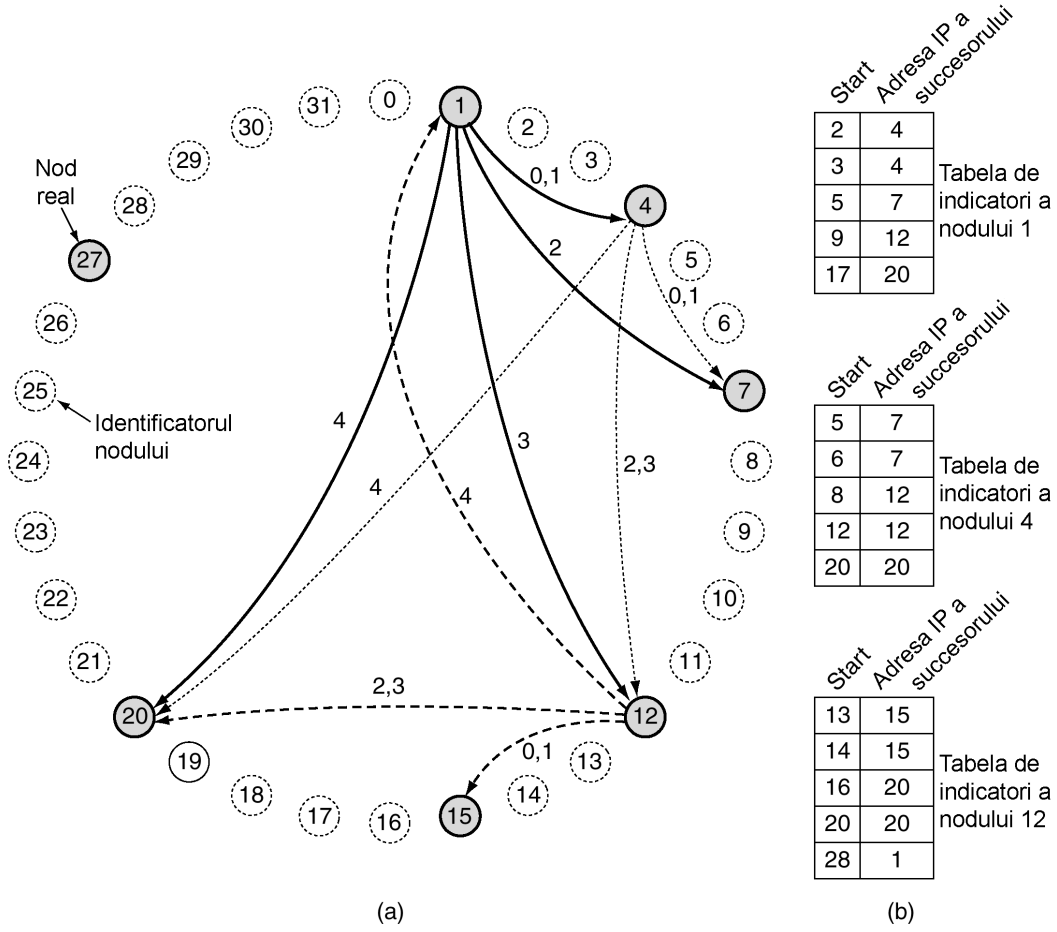


Fig. 5-24. (a) Un set de 32 de identificatori de noduri aranjați în cerc. Nodurile hașurate corespund mașinilor existente. Arcele arată indicatorii de la nodurile 1, 4 și 12. Etichetele arcelor reprezintă indici în tabele. (b) Exemple de tabele de indicatori.

Ca o primă optimizare, fiecare nod ar putea memora adresele IP ale succesorului și predecesorului, astfel încât interogările ar putea fi trimise fie în sensul acelor de ceasornic, fie în sens invers, pe calea considerată mai scurtă. De exemplu, nodul 7 din fig. 5-24 poate să trimită în sensul acelor de ceasornic pentru a găsi nodul cu identificatorul 10, dar în sens invers pentru a găsi nodul cu identificatorul 3.

Chiar și cu două posibilități pentru direcție, căutarea liniară prin toate nodurile este foarte ineficientă într-un sistem punct la punct mare, deoarece numărul mediu de noduri implicat într-o căutare este de *n/2*. Pentru a mări considerabil viteza de căutare, fiecare nod memorează de asemenea ceea ce Chord numește **tabelă de indicatori (finger table)**. Tabela de indicatori are *m* intrări, indexate de la 0 la *m-1,* fiecare indicând către un nod real diferit. Fiecare dintre intrări are două câmpuri: *start* și adresa IP a *successor(start)*, așa cum este prezentat pentru cele trei noduri din fig. 5-24(b). Valoarea acestor câmpuri pentru intrarea *i* a nodului *k* sunt:

start = $k + 2^i$ *(modulo* 2^m*)*
Adresa IP a successor(start[i])

Observați că fiecare nod memorează adresa IP a unui număr relativ mic de noduri și că majoritatea acestora sunt foarte apropiate din punctul de vedere al identificatorilor de noduri.

Folosind tabela de indicatori, căutarea *cheii* la nodul *k* se desfășoară după cum urmează. Dacă *cheia* este între *k* și *successor*(*k*), atunci nodul care deține informația despre *cheie* este *succesor(k)* și căutarea se termină. Altfel, se caută în tabela de indicatori pentru a găsi intrarea al cărui câmp *start* este predecesorul cel mai apropiat al *cheii.* Apoi se trimite direct la adresa IP din intrarea din tabela de indicatori o cerere în care se cere continuarea căutării. Deoarece aceasta este mai apropiată de cheie, dar inferioară, sunt mari șanse să fie capabilă să returneze un răspuns după un număr mic de interogări suplimentare. De fapt, deoarece fiecare căutare înjumătățește distanța rămasă până la țintă, se poate demonstra că numărul mediu de căutări este $log_2 n$.

Ca prim exemplu, să considerăm căutarea *key* = 3 la nodul 1. Deoarece nodul 1 știe că 3 se află între el și succesorul său, 4, nodul dorit este 4 și căutarea s-a terminat, returnându-se adresa IP a nodului 4.

Ca un al doilea exemplu, să considerăm căutarea *key* = 14 la nodul 1. Deoarece 14 nu este între 1 și 4, este analizată tabela de indicatori. Cel mai apropiat predecesor al lui 14 este 9, așa că cererea este trimisă către adresa IP din intrarea 9, și anume, cea a nodului 12. Nodul 12 vede că 14 se află între el și successor(15), așa că returnează adresa IP a nodului 15.

Ca un al treilea exemplu, să considerăm căutarea *key* = 16 la nodul 1. Din nou se trimite o interogare la nodul 12, dar de această dată nodul 12 nu cunoaște răspunsul. Caută nodul cel mai apropiat predecesor al lui 16 și îl găsește pe 14, care furnizează adresa IP a nodului 15. Acestuia îi este trimisă o interogare. Nodul 15 observă că 16 se află între el și succesorul său (20), așa că returnează apelantului adresa IP a lui 20, care se întoarce către nodul 1.

Deoarece nodurile apar și dispar mereu, Chord trebuie să trateze cumva aceste operații. Presupunem că atunci când sistemul a început să funcționeze, el era îndeajuns de mic pentru ca nodurile să poată schimba informații direct, pentru a construi primul cerc și tabelele de indicatori. După aceea este necesară o procedură automată, după cum urmează. Când un nod nou, *r*, vrea să se alăture, trebuie să contacteze un nod existent și să-i ceară să caute adresa IP a successor(r). După aceea noul nod îl întreabă pe successor(r) cine este predecesorul său. Apoi noul nod cere ambelor noduri să îl insereze pe *r* între ele în cerc. De exemplu, dacă 24 din fig. 5-24 vrea să se alăture, roagă orice nod să caute *successor(24)*, care este 27. Apoi îl întreabă pe 27 cine este predecesorul său (20). După ce le

înștiințează pe amândouă de existența sa, 20 îl folosește pe 24 ca succesor, iar 27 îl folosește pe 24 ca predecesor. În plus, nodul 27 predă cheile din intervalul 21-24, care acum îi aparțin lui 24. În acest moment, 24 este inserat pe deplin.

Totuși, multe tabele de indicatori sunt acum greșite. Pentru a le corecta, fiecare nod rulează un proces în fundal care calculează periodic fiecare indicator prin apelarea funcției *successor*. Când una dintre aceste interogări ajunge la un nod nou, este actualizată intrarea corespunzătoare indicatorului.

Când un nod pleacă normal, predă cheile succesorului său și informează predecesorul de plecarea sa, astfel încât predecesorul poate să se lege la succesorul nodului care a plecat. Când se defectează un nod apare o problemă, deoarece predecesorul său nu are un succesor valid. Pentru a soluționa problema, fiecare nod păstrează date nu numai despre succesorul său direct, dar și despre *s* succesori direcți, pentru a-i permite să sară peste *s-1* noduri consecutive defecte și să se reconecteze la cerc.

Chord a fost folosit la construirea unui sistem de fișiere distribuit (Dabek ș.a.,2001) și alte aplicații, iar cercetările continuă. Un sistem punct la punct diferit, Pastry, și aplicațiile sale sunt descrise în (Rowstron și Druschel, 2001a; și Rowstron și Druschel, 2001b). Un al treilea sistem punct la punct, Freenet, este discutat în (Clarke ș.a., 2002). Un al patrulea sistem de acest tip este descris în (Ratnasamy ș.a., 2001).

5.3 ALGORITMI PENTRU CONTROLUL CONGESTIEI

Atunci când foarte multe pachete sunt prezente într-o subrețea (sau o parte a unei subrețele), performanțele se degradează. Această situație se numește **congestie** (**congestion**). Fig. 5-25 prezintă simptomele. Când numărul de pachete emise în subrețea de calculatoarele gazdă nu depășește capacitatea de transport, ele sunt livrate integral (cu excepția celor care sunt afectate de erori de transmisie), iar numărul celor livrate este proporțional cu numărul celor emise. Totuși, atunci când traficul crește prea mult, ruterele încep să nu mai facă față și să piardă pachete. Aceasta tinde să înrăutățească lucrurile. La un trafic foarte intens performanțele se deteriorează complet și aproape nici un pachet nu mai este livrat.

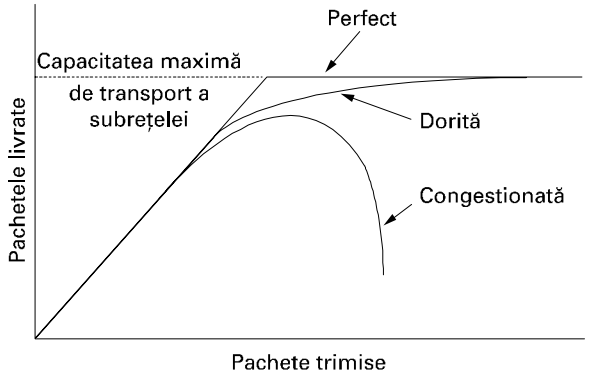


Fig. 5-25. Dacă traficul este prea intens, apare congestia și performanțele se degradează puternic.

Congestia poate fi produsă de mai mulți factori. Dacă dintr-o dată încep să sosească șiruri de pachete pe trei sau patru linii de intrare și toate necesită aceeași linie de ieșire, atunci se va forma o coadă. Dacă nu există suficientă memorie pentru a le păstra pe toate, unele se vor pierde. Adăugarea de memorie poate fi folositoare până la un punct, dar Nagle (1987) a descoperit că dacă ruterele ar avea o cantitate infinită de memorie, congestia s-ar înrăutăți în loc să se amelioreze, deoarece până să ajungă la începutul cozii pachetele au fost deja considerate pierdute (timeout repetat) și s-au trimis duplicate. Toate aceste pachete vor fi trimise cu conștiinciozitate către următorul ruter, crescând încărcarea de-a lungul căii către destinație.

Și procesoarele lente pot cauza congestia. Dacă unitatea centrală (CPU) a ruterului este lentă în execuția funcțiilor sale (introducerea în cozi, actualizarea tabelelor etc.), se pot forma cozi, chiar dacă linia de comunicație nu este folosită la capacitate. Similar și liniile cu lățime de bandă scăzută pot provoca congestia. Schimbarea liniilor cu unele mai performante și păstrarea aceluiași procesor sau vice-versa de obicei ajută puțin, însă de cele mai multe ori doar deplasează punctul critic. De asemenea, îmbunătățirea parțială și nu totală a sistemului cel mai adesea mută punctul critic în altă parte. Adevărata problemă este de multe ori o incompatibilitate între părți ale sistemului. Ea va persista până ce toate componentele sunt în echilibru.

Este important să subliniem diferența dintre controlul congestiei și controlul fluxului, deoarece relația este subtilă. Controlul congestiei trebuie să asigure că subrețeaua este capabilă să transporte întreg traficul implicat. Este o problemă globală, implicând comportamentul tuturor calculatoarelor gazdă, al tuturor ruterelor, prelucrarea de tip memorează-și-retransmite (store-and-forward) din rutere și toți ceilalți factori care tind să diminueze capacitatea de transport a subrețelei.

Controlul fluxului, prin contrast, se referă la traficul capăt la capăt între un expeditor și un destinatar. Rolul său este de a împiedica un expeditor rapid să trimită date continuu, la o viteză mai mare decât cea cu care destinatarul poate consuma datele. Controlul fluxului implică frecvent existența unui feed-back de la receptor către emițător, pentru a spune emițătorului cum se desfășoară lucrurile la celălalt capăt.

Pentru a vedea diferența dintre aceste două concepte, să considerăm o rețea cu fibre optice cu o capacitate de 1000 gigabiți/sec pe care un supercalculator încearcă să transfere un fișier către un calculator personal la 1 Gbps. Deși nu există nici o congestie (rețeaua nu are nici un fel de probleme), controlul fluxului este necesar pentru a forța supercalculatorul să se oprească des, pentru a permite calculatorului personal să și respire.

La cealaltă extremă, să considerăm o rețea de tip memorează-și-retransmite (store-and-forward), cu linii de 1 Mbps și 1000 de calculatoare mari, din care jumătate încearcă să transfere fișiere la 100 Kbps către cealaltă jumătate. Aici problema nu apare datorită unui emițător rapid care surclasează un receptor lent, ci pentru că traficul cerut depășește posibilitățile rețelei.

Motivul pentru care controlul congestiei și controlul fluxului sunt adesea confundate este acela că unii algoritmi pentru controlul congestiei funcționează trimițând mesaje înapoi către diferitele surse, spunându-le să încetinească atunci când rețeaua are probleme. Astfel, un calculator gazdă poate primi un mesaj de încetinire fie din cauză că receptorul nu suportă încărcarea, fie pentru că rețeaua este depășită. Vom reveni asupra acestui punct mai târziu.

Vom începe studiul algoritmilor pentru controlul congestiei prin studiul unui model general de tratare a acesteia. Apoi ne vom ocupa de principalele măsuri pentru prevenirea sa. După aceea, vom analiza o serie de algoritmi dinamici pentru tratarea congestiei odată ce a apărut.

5.3.1 Principii generale ale controlului congestiei

Multe din problemele care apar în sistemele complexe, cum ar fi rețelele de calculatoare, pot fi privite din punctul de vedere al unei teorii a controlului. Această abordare conduce la împărțirea tuturor soluțiilor în două grupe: în buclă deschisă și în buclă închisă. Soluțiile în buclă deschisă încearcă să rezolve problema printr-o proiectare atentă, în esență să se asigure că problema nu apare. După ce sistemul este pornit și funcționează, nu se mai fac nici un fel de corecții.

Instrumentele pentru realizarea controlului în buclă deschisă decid când să se accepte trafic nou, când să se distrugă pachete și care să fie acestea, realizează planificarea deciziilor în diferite puncte din rețea. Toate acestea au ca numitor comun faptul că iau decizii fără a ține cont de starea curentă a rețelei.

Prin contrast, soluțiile în buclă închisă se bazează pe conceptul de reacție inversă (feedback loop). Această abordare are trei părți, atunci când se folosește pentru controlul congestiei:

1. Monitorizează sistemul pentru a detecta când și unde se produce congestia.
2. Trimite aceste informații către locurile unde se pot executa acțiuni.
3. Ajustează funcționarea sistemului pentru a corecta problema.

În vederea monitorizării subrețelei pentru congestie se pot folosi diverse de metrici. Cele mai utilizate sunt procentul din totalul pachetelor care au fost distruse din cauza lipsei spațiului temporar de memorare, lungimea medie a cozilor de așteptare, numărul de pachete care sunt retransmise pe motiv de timeout, întârzierea medie a unui pachet, deviația standard a întârzierii unui pachet. În toate cazurile, valorile crescătoare indică creșterea congestiei.

Al doilea pas în bucla de reacție este transferul informației legate de congestie de la punctul în care a fost depistată la punctul în care se poate face ceva. Varianta imediată presupune trimiterea unor pachete de la ruterul care a detectat congestia către sursa sau sursele de trafic, pentru a raporta problema. Evident, aceste pachete suplimentare cresc încărcarea rețelei exact la momentul în care acest lucru era cel mai puțin dorit, subrețeaua fiind congestionată.

Există însă și alte posibilități. De exemplu, poate fi rezervat un bit sau un câmp în fiecare pachet, pentru a fi completat de rutere dacă congestia depășește o anumită valoare de prag. Când un ruter detectează congestie, el completează câmpurile tuturor pachetelor expediate, pentru a-și preveni vecinii.

O altă abordare este ca ruterele sau calculatoarele gazdă să trimită periodic pachete de probă pentru a întreba explicit despre congestie. Aceste informații pot fi apoi folosite pentru a ocoli zonele cu probleme. Unele stații de radio au elicoptere care zboară deasupra orașelor pentru a raporta congestiile de pe drumuri și a permite ascultătorilor mobili să își dirijeze pachetele (mașinile) astfel încât să ocolească zonele fierbinți.

În toate schemele cu feedback se speră că informarea asupra producerii congestiei va determina calculatoarele gazdă să ia măsurile necesare pentru a reduce congestia. Pentru ca o schemă să funcționeze corect, duratele trebuie reglate foarte atent. Dacă un ruter strigă STOP de fiecare dată când sosesc două pachete succesive și PLEACĂ (eng.: GO) de fiecare dată când este liber mai mult de 20 μsec, atunci sistemul va oscila puternic și nu va converge niciodată. Pe de altă parte, dacă va aștepta 30 de minute pentru a fi sigur înainte de a spune ceva, mecanismul pentru controlul congestiei reacționează prea lent pentru a fi de vreun folos real. Pentru a funcționa corect sunt necesare unele medieri, dar aflarea celor mai potrivite constante de timp nu este o treabă tocmai ușoară.

Se cunosc numeroși algoritmi pentru controlul congestiei. Pentru a oferi o modalitate de organizare a lor, Yang și Reddy (1995) au dezvoltat o taxonomie pentru algoritmii de control al congestiei. Ei încep prin a împărți algoritmii în cei în buclă deschisă și cei în buclă închisă, așa cum s-a precizat anterior. În continuare împart algoritmii cu buclă deschisă în unii care acționează asupra sursei și alții care acționează asupra destinației. Algoritmii în buclă închisă sunt de asemenea împărțiți în două subcategorii, cu feedback implicit și cu feedback explicit. În algoritmii cu feedback explicit, pachetele sunt trimise înapoi de la punctul unde s-a produs congestia către sursă, pentru a o avertiza. În algoritmii impliciți, sursa deduce existența congestiei din observații locale, cum ar fi timpul necesar pentru întoarcerea confirmărilor.

Prezența congestiei înseamnă că încărcarea (momentană) a sistemului este mai mare decât cea pe care o pot suporta resursele (unei părți a sistemului). Pentru rezolvare vin imediat în minte două soluții: sporirea resurselor sau reducerea încărcării. De exemplu subrețeaua poate începe să folosească linii telefonice pentru a crește temporar lățimea de bandă între anumite puncte. În sistemele bazate pe sateliți, creșterea puterii de transmisie asigură de regulă creșterea lățimii de bandă. Spargerea traficului pe mai multe căi în locul folosirii doar a celei mai bune poate duce efectiv la creșterea lățimii de bandă. În fine, ruterele suplimentare, folosite de obicei doar ca rezerve pentru copii de siguranță (backups) (pentru a face sistemul tolerant la defecte), pot fi folosite pentru a asigura o capacitate sporită atunci când apar congestii serioase.

Totuși, uneori nu este posibilă creșterea capacității sau aceasta a fost deja crescută la limită. Atunci singura cale de a rezolva congestia este reducerea încărcării. Sunt posibile mai multe metode pentru reducerea încărcării, cum ar fi refuzul servirii anumitor utilizatori, degradarea serviciilor pentru o parte sau pentru toți utilizatorii și planificarea cererilor utilizatorilor într-o manieră mai previzibilă.

Unele dintre aceste metode, pe care le vom studia pe scurt, pot fi aplicate cel mai bine circuitelor virtuale. Pentru subrețelele care folosesc intern circuite virtuale aceste metode pot fi utilizate la nivelul rețea. Pentru subrețele bazate pe datagrame ele pot fi totuși folosite uneori pentru conexiuni la nivelul transport. În acest capitol, ne vom concentra pe folosirea lor în cadrul nivelului rețea. În următorul, vom vedea ce se poate face la nivelul transport pentru a controla congestia.

5.3.2 Politici pentru prevenirea congestiei

Să începem studiul asupra metodelor pentru controlul congestiei prin analiza sistemelor cu buclă deschisă. Aceste sisteme sunt proiectate astfel încât să minimizeze congestia, în loc să o lase să se producă și apoi să reacționeze. Ele încearcă să-și atingă scopul folosind politici corespunzătoare, la diferite niveluri. În fig. 5-26 sunt prezentate diferite politici pentru nivelul legătură de date, rețea și transport, care pot influența congestia (Jain, 1990).

Să începem cu nivelul legătură de date și să ne continuăm apoi drumul spre niveluri superioare. Politica de retransmisie stabilește cât de repede se produce timeout la un emițător și ce transmite acesta la producerea timeout-ului. Un emițător vioi, care produce repede timeout și retransmite toate pachetele în așteptare folosind retrimiterea ultimelor *n,* va produce o încărcare mai mare decât un emițător calm, care folosește retrimiterea selectivă. Strâns legată de acestea este politica de memorare în zonă tampon. Dacă receptorii distrug toate pachetele în afara secvenței, acestea vor trebui retransmise ulterior, introducând o încărcare suplimentară. În ceea ce privește controlul congestiei, repetarea selectivă este, în mod sigur, mai bună decât retrimiterea ultimelor *n*.

Nivel	Politică
Transport	Politica de retransmisie Politica de memorare temporară a pachetelor în afară de secvență (out-of-order caching) Politica de confirmare Politica de control al fluxului Determinarea timeout-ului
Rețea	Circuite virtuale contra datagrame în interiorul subrețelei Plasarea pachetelor în cozi de așteptare și politici de servire Politica de distrugere a pachetelor Algoritmi de dirijare Gestiunea timpului de viață al pachetelor
Legătură de date	Politica de retransmitere Politica de memorare temporară a pachetelor în afară de secvență (out-of-order caching) Politica de confirmare Politica de control al fluxului

Fig. 5-26. Politici care influențează congestia.

Și politica de confirmare afectează congestia. Dacă fiecare pachet este confirmat imediat, pachetele de confirmare vor genera un trafic suplimentar. Totuși, în cazul în care confirmările sunt preluate de traficul de răspuns, se pot produce timeout-uri și retransmisii suplimentare. O schemă prea strânsă pentru controlul fluxului (de exemplu fereastră mică) reduce volumul de date și ajută în lupta cu congestia.

La nivelul rețea, alegerea între folosirea circuitelor virtuale și datagrame influențează congestia, deoarece mulți algoritmi pentru controlul congestiei funcționează doar pe subrețele cu circuite virtuale. Plasarea în cozi de așteptare a pachetelor și politicile de servire specifică dacă ruterele au o coadă pentru fiecare linie de intrare, o coadă pentru fiecare linie de ieșire sau ambele. Mai precizează ordinea în care se prelucrează pachetele (de exemplu round robin sau bazată pe priorități). Politica de distrugere a pachetelor este regula care stabilește ce pachet este distrus dacă nu mai există spațiu. O politică bună va ajuta la eliminarea congestiei, pe când una greșită o va accentua.

Un algoritm de dirijare bun poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor, în timp ce un algoritm neperformant ar putea trimite toate pachetele pe aceeași linie, care deja este congestionată. În fine, gestiunea timpului de viață asociat pachetelor stabilește cât de mult poate trăi un pachet înainte de a fi distrus. Dacă acest timp este prea mare, pachetele pierdute vor încurca pentru mult timp activitatea, iar dacă este prea mic, este posibil să se producă timeout înainte de a ajunge la destinație, provocând astfel retransmisii.

La nivelul transport apar aceleași probleme ca la nivelul legăturii de date dar, în plus, determinarea intervalului de timeout este mai dificil de realizat, deoarece timpul de tranzit prin rețea este mai greu de prezis decât timpul de tranzit pe un fir între două rutere. Dacă intervalul de timeout este prea mic, vor fi trimise inutil pachete suplimentare. Dacă este prea mare, congestia se va reduce, însă timpul de răspuns va fi afectat de pierderea unui pachet.

5.3.3 Controlul congestiei în subrețelele bazate pe circuite virtuale

Metodele pentru controlul congestiei descrise anterior sunt în principiu în buclă deschisă: ele încearcă, în primul rând, să prevină apariția congestiei, în loc să o trateze după ce a apărut. În această

secțiune, vom descrie unele abordări ale controlului dinamic al congestiei în subrețelele bazate pe circuite virtuale. În următoarele două, vom studia tehnici ce pot fi folosite în orice subrețea.

O tehnică larg răspândită pentru a împiedica agravarea unei congestii deja apărute este **controlul admisiei**. Ideea este simplă: odată ce s-a semnalat apariția congestiei, nu se mai stabilesc alte circuite virtuale până ce problema nu s-a rezolvat. Astfel, încercarea de a stabili o nouă conexiune la nivel transport eșuează. Lăsând tot mai mulți utilizatori să stabilească conexiuni, nu ar face decât să agraveze lucrurile. Deși această abordare este dură, ea este simplă și ușor de realizat. În sistemul telefonic, dacă o centrală devine supraaglomerată, ea practică controlul admisiei, nemaifurnizând tonul.

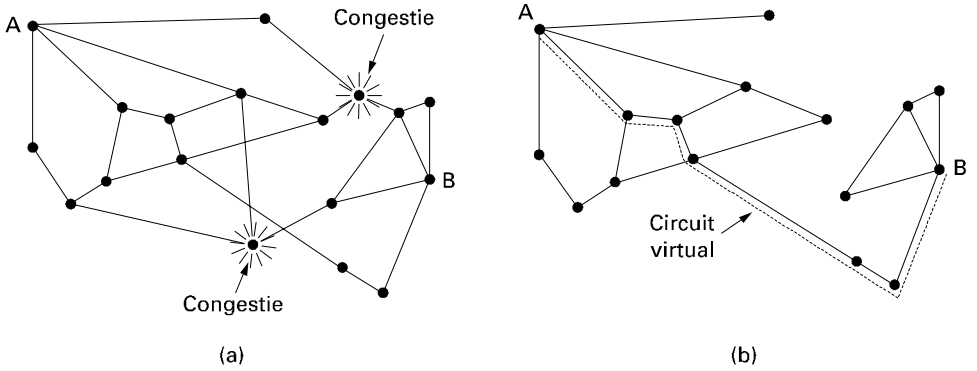


Fig. 5-27. (a) O subrețea congestionată. (b) O subrețea reconfigurată care elimină congestia. Se prezintă și un circuit virtual de la *A* la *B*.

O alternativă este de a permite stabilirea de noi circuite virtuale, dar de a dirija cu atenție aceste noi circuite, ocolind zonele cu probleme. De exemplu, să considerăm subrețeaua din fig. 5-27(a), în care două rutere sunt congestionate, așa cum se poate observa.

Să presupunem că un calculator gazdă atașat ruterului *A* dorește să stabilească o conexiune cu un altul, atașat ruterului *B*. În mod normal, această conexiune ar trece prin unul dintre cele două rutere congestionate. Pentru a evita această situație, putem redesena subrețeaua așa cum se arată în fig. 5-27(b), omițând ruterele congestionate și toate liniile asociate. Linia punctată arată o posibilă cale pentru circuitul virtual, care evită ruterele congestionate.

O altă strategie specifică circuitelor virtuale este negocierea unei înțelegeri între calculatorul gazdă și subrețea la stabilirea unui circuit virtual. Această înțelegere specifică în mod normal volumul și forma traficului, calitatea serviciului cerut și alți parametri. De obicei, pentru a-și respecta partea din înțelegere, subrețeaua își rezervă resurse de-a lungul căii în momentul stabilirii circuitului virtual. Resursele pot include spațiu pentru tabele și zone tampon în rutere și lățime de bandă pe linii. În acest fel, congestia nu prea are șanse să se producă pe noul circuit virtual, deoarece toate resursele sunt garantate a fi disponibile.

Acest tip de rezervare poate fi aplicat tot timpul ca o procedură standard sau doar atunci când rețeaua este congestionată. Dezavantajul încercării de a-l face tot timpul este risipa de resurse. Dacă șase circuite virtuale care pot folosi 1 Mbps trec toate prin aceeași legătură fizică de 6 Mbps, linia trebuie marcată ca plină, chiar dacă este puțin probabil ca toate cele șase circuite să transmită la nivel maxim în același timp. În consecință, prețul controlului congestiei este lățime de bandă nefolosită (adică risipită) în cazuri normale.

5.3.4 Controlul congestiei în subrețele datagramă

Să ne oprim acum asupra unor abordări care pot fi folosite în subrețelele bazate pe datagrame (dar și în cele bazate pe circuite virtuale). Fiecare ruter poate gestiona cu ușurință folosirea liniilor sale de ieșire precum și alte resurse. De exemplu, el poate asocia fiecărei linii o variabilă reală, u, a cărei valoare, între 0.0 și 1.0, reflectă utilizarea recentă a acelei linii. Pentru a menține o estimare cât mai bună a lui u, se poate face periodic eșantionarea utilizării instantanee a liniei f (fie 0, fie 1) și apoi actualizarea lui u astfel:

$$u_{new} = au_{old} + (1 - a)f$$

unde constanta a determină cât de repede uită ruterul istoria recentă.

Ori de câte ori u depășește pragul, linia de ieșire intră într-o stare de „avertisment". Fiecare pachet nou-venit este verificat pentru a se vedea dacă linia de ieșire asociată este în starea de avertisment. Dacă este așa, atunci se iau măsuri. Acțiunea executată poate fi una dintr-o mulțime de alternative, pe care le vom discuta acum.

Bitul de avertizare

Vechea arhitectură DECNET semnala starea de avertizare prin setarea unui bit special în antetul pachetului. La fel procedează și retransmisia cadrelor (frame relay). Când pachetul ajunge la destinația sa, entitatea transport copiază bitul în următoarea confirmare trimisă înapoi la sursă. În acel moment, sursa își reduce traficul.

Atât timp cât ruterul a fost în starea de avertizare, acesta a continuat să seteze bitul de avertizare, ceea ce înseamnă că sursa a primit în continuare confirmări cu acest bit setat. Sursa a monitorizat fracțiunea de confirmări cu bitul setat și și-a ajustat rata de transmisie corespunzător. Atâta timp cât au continuat să sosească biți de avertizare, sursa și-a micșorat continuu rata de transmisie. Odată cu încetinirea sosirii acestora, sursa și-a mărit rata de transmisie. De observat că, deoarece fiecare ruter de-a lungul căi ar fi putut seta bitul de avertizare, traficul a crescut numai atunci când nici unul dintre rutere nu a avut probleme.

Pachete de șoc

Algoritmul anterior pentru controlul congestiei este destul de ingenios. Folosește o modalitate ocolită de a înștiința sursa să încetinească transmisia. De ce nu-i spune direct? În această abordare, ruterul trimite un **pachet de șoc** către calculatorul gazdă sursă, dându-i destinația găsită în pachet. Pachetul original este marcat (un bit din antet este comutat) pentru a nu se mai genera pachete de șoc pe calea aleasă, apoi este retrimis în mod obișnuit.

Un calculator gazdă sursă care primește un pachet de șoc trebuie să reducă traficul trimis spre destinația specificată cu X procente. Deoarece alte pachete trimise către aceeași destinație sunt deja pe drum și vor genera alte pachete de șoc, calculatorul gazdă ar trebui să ignore pachetele de șoc referitoare la destinația respectivă o anumită perioadă de timp. După ce perioada s-a scurs, calculatorul gazdă așteaptă alte pachete de șoc un alt interval. Dacă sosește un astfel de pachet, linia este încă congestionată, astfel încât calculatorul va reduce fluxul și mai mult și va reîncepe să ignore pachetele de șoc. Dacă pe perioada de ascultare nu sosesc pachete de șoc, atunci calculatorul gazdă poate să crească din nou fluxul. Reacția implicită a acestui protocol poate ajuta la prevenirea congestiei, neștrangulând fluxul decât dacă au apărut probleme.

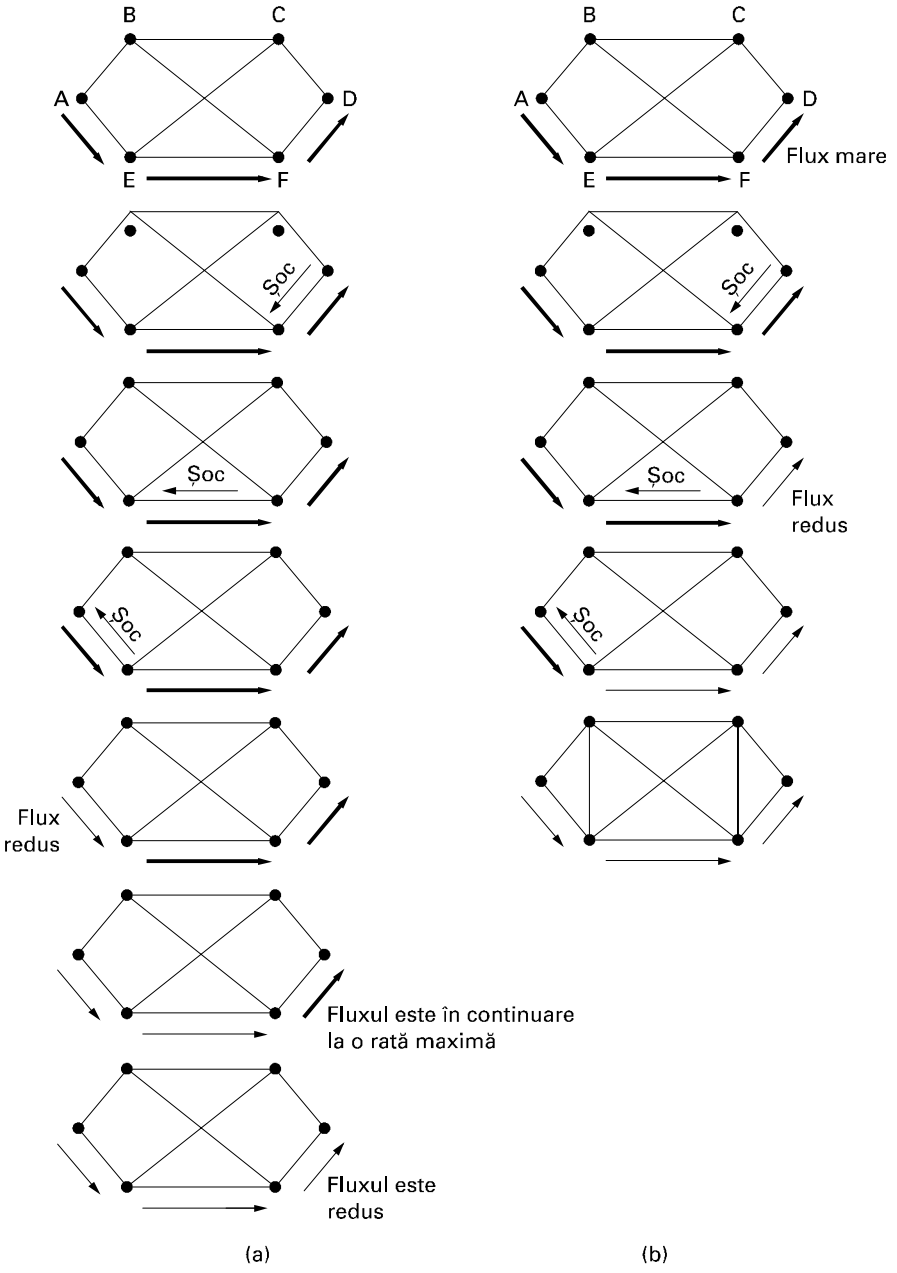


Fig. 5-28. (a) Un pachet de șoc care afectează doar sursa. (b) Un pachet de șoc care afectează fiecare salt prin care trece.

Calculatorul gazdă poate reduce traficul prin ajustarea parametrilor asociați politicii folosite, de exemplu dimensiunea ferestrei. De obicei, primul pachet de șoc determină scăderea ratei datelor la

0.50 din valoarea anterioară, următoarea reduce traficul la 0.25 și așa mai departe. Creșterea valorilor are loc cu rate mai mici pentru a preveni reinstalarea rapidă a congestiei.

Au fost propuse mai multe variante ale acestui algoritm pentru controlul congestiei. În una dintre acestea, fiecare ruter poate menține mai multe valori de referință (praguri). În funcție de pragul depășit, pachetul de șoc poate conține un avertisment blând, unul sever sau un ultimatum.

Altă variantă prevede folosirea lungimilor cozilor sau a utilizării zonelor tampon în locul utilizării liniilor, ca semnal de comutare. Evident, pentru această metrică se poate folosi aceeași ponderare exponențială ca și pentru u.

Pachete de șoc salt cu salt

La viteze mari sau pe distanțe mari, trimiterea unui pachet de șoc către calculatorul sursă nu funcționează normal, reacția fiind întârziată. Să considerăm, de exemplu, un calculator în San-Francisco (ruterul A din fig. 5-28) care trimite trafic către un calculator din New York (ruterul D din fig. 5-28) la 155 Mbps. Dacă ruterul din New York rămâne fără zone tampon, atunci pachetului de șoc îi vor trebui circa 30 msec pentru ca să revină la San Francisco solicitând încetinirea. Propagarea pachetului de șoc este prezentată ca al doilea, al treilea și al patrulea pas din fig. 5-28(a). În aceste 30 msec, se pot trimite 4.6 MB. Chiar atunci când calculatorul gazdă din San Franciso se oprește imediat, cei 4.6 MB din conductă (eng.: pipe) vor continua să sosească și trebuie luați în seamă. De-abia în a șaptea diagramă din fig. 5-28(a) ruterul din New York va observa reducerea fluxului.

O abordare alternativă este ca pachetele de șoc să aibă efect în fiecare salt prin care trec, așa cum se arată în secvența din fig. 5-28(b). În acest caz, de îndată ce pachetul de șoc ajunge la F, F trebuie să reducă fluxul spre D. Procedând în acest fel, se cere din partea lui F alocarea mai multor zone tampon pentru flux, în timp ce sursa continuă să emită la viteză maximă, însă D simte imediat o ușurare, la fel ca într-o reclamă TV pentru înlăturarea durerii de cap. La pasul următor, pachetul de șoc va ajunge la E, spunându-i lui E să reducă fluxul spre F. Această acțiune solicită puternic zonele tampon ale lui E, însă are un efect benefic imediat asupra lui F, ușurându-i munca. În fine, pachetul de șoc ajunge la A, și fluxul scade efectiv.

Efectul acestei scheme salt cu salt asupra rețelei este asigurarea unei eliberări imediate la locul congestiei cu prețul folosirii mai multor zone tampon. În acest fel congestia poate fi înăbușită imediat ce se manifestă, fără a se pierde nici un pachet. Ideea este discutată în amănunt în (Mishra și Kanakia, 1992), unde se furnizează și rezultatele unei simulări.

5.3.5 Împrăștierea încărcării

Când nici una dintre metodele anterioare nu reduc congestia, ruterele pot să-și scoată la bătaie artileria grea: împrăștierea încărcării. **Împrăștierea încărcării** este un mod simpatic de a spune că atunci când ruterele sunt inundate de pachete pe care nu le mai pot gestiona, pur și simplu le aruncă. Termenul provine din domeniul distribuției energiei electrice, unde se referă la practica serviciilor publice care lasă intenționat fără energie electrică anumite zone, pentru a salva întreaga rețea de la colaps, în zilele călduroase de vară când cererea de energie electrică depășește puternic oferta.

Un ruter care se îneacă cu pachete poate alege la întâmplare pachetele pe care să le arunce, însă de obicei el poate face mai mult decât atât. Alegerea pachetelor care vor fi aruncate poate depinde de aplicațiile care rulează. Pentru transferul de fișiere, un pachet vechi este mai valoros decât unul nou, deoarece aruncarea pachetului 6 și păstrarea pachetelor de la 7 la 10 va cauza producerea unei „spărturi” în fereastra receptorului, care poate forța retrimiterea

pachetelor de la 6 la 10 (dacă receptorul distruge automat pachetele care sunt în afara secvenței). În cazul unui fișier de 12 pachete, aruncarea pachetului 6 poate necesita retransmiterea pachetelor de la 7 la 12, în timp ce aruncarea lui 10 va necesita doar retrimiterea pachetelor de la 10 la 12. În contrast, pentru multimedia, un pachet nou este mult mai important decât un pachet vechi. Prima politică (vechiul este mai bun decât noul) este numită adesea **a vinului**, iar ultima (noul este mai bun decât vechiul) este numită **a laptelui**.

Un pas inteligent mai departe presupune cooperare din partea expeditorilor. Pentru multe aplicații, unele pachete sunt mai importante decât altele. De exemplu, unii algoritmi de compresie a imaginilor transmit periodic un cadru întreg și apoi trimit următoarele cadre ca diferențe față de ultimul cadru complet. În acest caz, aruncarea unui pachet care face parte dintr-o diferență este de preferat aruncării unuia care face parte dintr-un cadru întreg. Ca un alt exemplu, să considerăm trimiterea documentelor care conțin text ASCII și imagini. Pierderea unei linii de pixeli dintr-o imagine este de departe mai puțin dăunătoare decât pierderea unei linii dintr-un text.

Pentru a implementa o politică inteligentă de distrugere a pachetelor, aplicațiile trebuie să înscrie pe fiecare pachet clasa de prioritate din care face parte, pentru a indica cât de important este. Dacă ele fac aceasta, atunci când trebuie distruse unele pachete, ruterele vor începe cu cele din clasa cea mai slabă, vor continua cu cele din următoarea clasă și așa mai departe. Evident, dacă nu ar fi nici un stimulent pentru a marca pachetele și altfel decât FOARTE IMPORTANT - A NU SE DISTRUGE NICIODATĂ, SUB NICI UN MOTIV, nimeni nu ar face acest lucru.

Stimulentul ar putea fi sub formă de bani, adică pachetele mai puțin prioritare să fie mai ieftin de trimis decât cele cu prioritate mare. Ca alternativă, expeditorilor li s-ar putea permite să trimită pachete cu priorități mari cu condiția ca încărcarea să fie mică, dar odată cu creșterea încărcării acestea ar fi aruncate, încurajând astfel utilizatorii să înceteze trimiterea lor.

O altă opțiune ar fi să se permită calculatoarelor gazdă să depășească limitele specificate în contractul negociat la inițializarea circuitului virtual (să folosească o lățime de bandă mai mare decât li s-a permis), dar cu condiția ca tot traficul în exces să fie marcat ca fiind de prioritate scăzută. O astfel de strategie nu este de fapt o idee rea, pentru că utilizează în mod mai eficient resursele mai puțin folosite, permițând astfel calculatoarelor gazdă să le utilizeze atâta timp cât nimeni altcineva nu este interesat, dar fără să stabilească dreptul la ele atunci când situația devine mai dură.

Detecția aleatoare timpurie

Este bine cunoscut faptul că a trata congestia imediat ce a fost detectată este mai eficient decât să o lași să-ți încurce treburile și apoi să încerci să te descurci cu ea. Această observație a dus la ideea de a arunca pachete înainte ca toată zona tampon să fie într-adevăr epuizată. Un algoritm popular pentru a face acest lucru se numește **RED** (**Random Early Detection** – rom.: Detecția aleatoare timpurie) (Floyd și Jacobson, 1993). În unele protocoale de transport (incluzând TCP-ul), răspunsul la pachetele pierdute este ca sursa să încetinească. Raționamentul din spatele acestei logici este acela că TCP a fost proiectat pentru rețele cablate, iar acestea sunt foarte sigure, astfel încât pachetele pierdute se datorează mai degrabă depășirii spațiului tampon decât erorilor de transmisie. Acest lucru poate fi exploatat pentru a ajuta la reducerea congestiei.

Prin aruncarea pachetelor de către rutere înainte ca situația să devină fără speranță (de unde și termenul “timpuriu” din denumire), ideea este de a lua măsuri înainte de a fi prea târziu. Pentru a determina când să înceapă aruncarea, ruterele mențin neîntrerupt o medie a lungimilor cozilor lor. Când lungimea medie a cozii unei linii depășește o limită, se spune că linia este congestionată și se iau măsuri.

Având în vedere că ruterul probabil că nu poate spune care sursă produce necazul cel mai mare, alegerea la întâmplare a unui pachet din coada care a determinat această acțiune este probabil cel mai bun lucru pe care îl poate face.

Cum ar trebui ruterul să informeze sursa despre problemă? O cale este aceea de a-i trimite un pachet de șoc, după cum am descris. O problemă a acestei abordări este că încarcă și mai mult rețeaua deja congestionată. O strategie diferită este să arunce pur și simplu pachetul selectat și să nu raporteze acest lucru. Sursa va observa în cele din urmă lipsa confirmării și va lua măsuri. Deoarece știe că pachetele pierdute sunt în general determinate de congestie și aruncări, va reacționa prin încetinire în loc să încerce mai abitir. Această formă implicită de reacție funcționează doar atunci când sursele răspund la pachetele pierdute prin reducerea ratei lor de transmisie. În rețele fără fir, unde majoritatea pierderilor se datorează zgomotelor legăturii aeriene, această abordare nu poate fi folosită.

5.3.6 Controlul fluctuațiilor

Pentru aplicații de genul transmisiilor audio sau video, nu prea contează dacă pachetele au nevoie de 20 msec sau de 30 msec pentru a fi distribuite, atât timp cât durata de tranzit este constantă. Variația (adică deviația standard) timpului de sosire a pachetului se numește **fluctuație.** O fluctuație mare, de exemplu dacă unele pachete au nevoie de 20 msec și altele de 30 msec pentru a ajunge, va produce sunet sau imagine de calitate inegală. Fluctuația este ilustrată în fig. 5-29. În contrast, o înțelegere ca 99 procente din pachete să fie livrate cu o întârziere de la 24.5 msec până la 25.5 msec ar putea fi acceptabilă.

Limita aleasă trebuie să fie, bineînțeles, reală. Ea trebuie să ia în calcul durata de tranzit a vitezei luminii și întârzierea minimă prin rutere și poate să-și lase o mică rezervă pentru unele întârzieri inevitabile.

Fluctuațiile pot fi limitate prin calcularea timpului de tranzit estimat pentru fiecare salt de-a lungul căii. Când un pachet ajunge la ruter, ruterul verifică să vadă cât de mult este decalat pachetul față de planificarea sa. Această informație este păstrată în pachet și actualizată la fiecare salt. Dacă pachetul a sosit înaintea planificării, el este ținut suficient pentru a reveni în grafic. Dacă pachetul este întârziat față de planificare, ruterul încearcă să-l trimită cât mai repede.

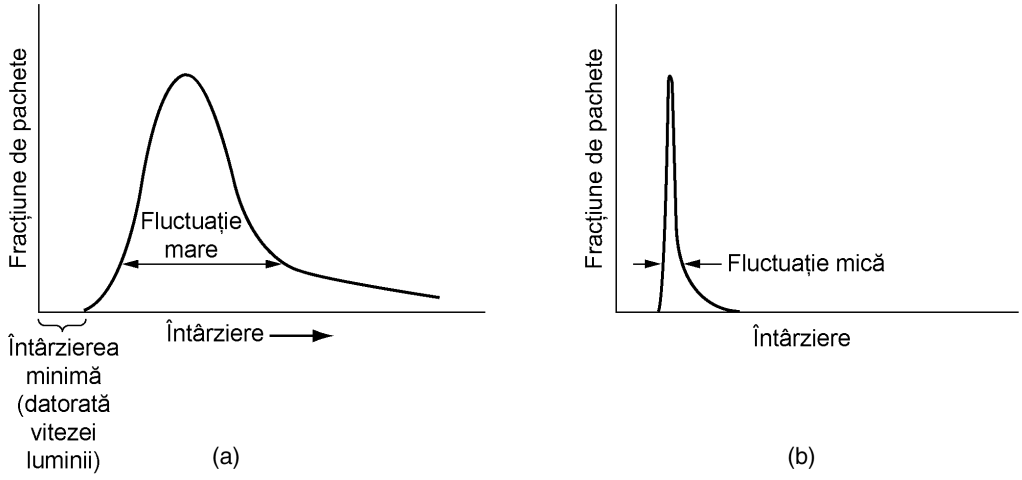


Fig. 5-29. (a) Fluctuație mare. (b) Fluctuație scăzută.

De fapt, algoritmul pentru determinarea pachetului care va merge mai departe dintr-un set de pachete care concurează pentru o linie de ieșire, va alege întotdeauna pachetul cu cea mai mare întârziere față de program. În acest fel, pachetele care au ajuns mai repede sunt încetinite, iar cele care sunt întârziate vor fi accelerate, în ambele cazuri reducându-se fluctuația.

În unele aplicații, cum ar fi aplicații video la cerere, fluctuațiile pot fi eliminate prin adăugarea unei zone tampon la receptor și apoi aducerea datelor pentru a fi afișate din zona tampon în loc de a fi luate din rețea în timp real. Totuși, pentru alte aplicații, în special acelea care necesită interacțiune în timp real între oameni, cum ar fi convorbirile telefonice și videoconferințele prin Internet, întârzierea inerentă utilizării zonelor tampon este inacceptabilă. Controlul congestiei este o arie activă a cercetării. Nivelul actual este rezumat în (Gevros et al., 2001).

5.4 CALITATEA SERVICIILOR

Tehnicile pe care le-am analizat în secțiunile precedente sunt proiectate pentru a reduce congestia și a îmbunătăți performanțele rețelei. Cu toate acestea, odată cu creșterea lucrului cu aplicații multimedia în rețea, de multe ori aceste măsuri luate ad-hoc nu sunt de ajuns. Este nevoie de preocupări serioase în proiectarea rețelelor și a protocoalelor pentru garantarea calității serviciilor. În secțiunile următoare ne vom continua studiul asupra performanței rețelei, dar acum ne vom concentra mai mult asupra modalităților de a furniza o calitate a serviciilor în conformitate cu necesitățile fiecărei aplicații. Ar trebui stipulat de la început faptul că, totuși, multe dintre aceste idei sunt în formare și sunt supuse schimbărilor.

5.4.1 Cerințe

Un șir de pachete de la o sursă la o destinație se numește **flux**. Într-o rețea orientată pe conexiune, toate pachetele aparținând unui flux merg pe aceeași rută; într-o rețea neorientată pe conexiune, acestea pot urma rute diferite. Necesitățile fiecărui flux pot fi caracterizate prin patru parametri primari: fiabilitatea, întârzierea, fluctuația și lățimea de bandă. Împreună, acestea determină **QoS** (**Quality of Service** – rom.: Calitatea serviciilor) pe care o necesită fluxul. În fig. 5-30 sunt listate câteva aplicații obișnuite precum și cerințele stringente ale acestora.

Aplicația	Fiabilitatea	Întârzierea	Fluctuația	Lățimea de bandă
Poșta electronică	Mare	Mică	Mică	Mică
Transfer de fișiere	Mare	Mică	Mică	Medie
Acces Web	Mare	Medie	Mică	Medie
Conectare la distanță	Mare	Medie	Medie	Mică
Audio la cerere	Mică	Mică	Mare	Medie
Video la cerere	Mică	Mică	Mare	Mare
Telefonie	Mică	Mare	Mare	Mică
Videoconferințe	Mică	Mare	Mare	Mare

Fig. 5-30. Cât de stringente sunt cerințele referitoare la calitatea serviciilor

Primele patru aplicații au cerințe stringente de fiabilitate. Nu se accepta livrarea de biți incorecți. Acest obiectiv este atins de obicei prin crearea unei sume de control pentru fiecare pachet și verifica-

rea acestei sume de control la destinație. Dacă un pachet este alterat pe parcurs, nu este confirmat și în cele din urmă va fi retransmis. Această strategie oferă o fiabilitate mare. Ultimele patru aplicații (audio/video) pot tolera erori, așa că nu este calculată sau verificată nici o sumă de control.

Aplicațiile de transfer de fișiere, inclusiv poșta electronică și video nu sunt sensibile la întârzieri. Dacă toate pachetele sunt întârziate uniform cu câteva secunde, nu este nici o problemă. Aplicațiile interactive, cum ar fi navigarea pe Web și conectarea la distanță, sunt mult mai sensibile la întârzieri. Aplicațiile de timp real, cum ar fi telefonia și videoconferințele au cerințe stricte de întârziere. Dacă toate cuvintele dintr-o convorbire telefonică sunt fiecare întârziate cu 2.000 secunde, utilizatorii vor găsi conexiunea inacceptabilă. Pe de altă parte, rularea de fișiere audio sau video de pe server nu cere întârzieri scăzute.

Primele trei aplicații nu sunt sensibile la sosirea pachetelor la intervale neregulate. Conectarea la distanță este într-un fel sensibilă la aceasta deoarece caracterele vor apărea pe ecran în rafale dacă conexiunea are multe fluctuații. Aplicațiile video și în special cele audio sunt extrem de sensibile la fluctuații. Dacă un utilizator urmărește un fișier video din rețea și cadrele sunt toate întârziate cu exact 2.000 secunde, nu este nici o problemă. Dar dacă timpul de transmisie variază aleator între 1 și 2 secunde, rezultatul va fi îngrozitor. Pentru audio, o fluctuație chiar și de câteva milisecunde este perfect sesizabilă.

În cele din urmă, aplicațiile diferă la cerințele de lățime de bandă, fără ca poșta electronică și conectarea la distanță să necesite o bandă prea largă, iar aplicațiile video de orice fel necesitând o bandă foarte largă. Rețelele ATM clasifică fluxul în patru mari categorii, în funcție de cerințele lor de QoS, după cum urmează:

1. Rată de transfer constantă (de exemplu telefonie).
2. Rată de transfer variabilă în timp real (de exemplu videoconferențiere compresată).
3. Rată de transfer variabilă, dar nu în timp real (de exemplu vizionarea unui film pe Internet).
4. Rată de transfer disponibilă (de exemplu transfer de fișiere).

Aceste categorii sunt de asemenea folositoare pentru alte scopuri și pentru alte tipuri de rețele. Rata de transfer constantă este o încercare de a simula un cablu, furnizând o lățime de bandă uniformă și o întârziere uniformă. Rata de transfer variabilă apare pentru video compresat, unele cadre fiind mai comprimate decât altele. Astfel, transmisia unui cadru cu o mulțime de detalii în el ar putea necesita transmisia mai multor biți, în timp ce transmisia imaginii unui perete alb s-ar putea comprima extrem de bine. Rata de transfer disponibilă este pentru aplicații care nu sunt sensibile la întârzieri sau la fluctuații, cum ar fi poșta electronică.

5.4.2 Tehnici pentru obținerea unei bune calități a serviciilor

Acum că știm câte ceva despre cerințele QoS, cum le obținem? Ei bine, pentru început, nu există o rețetă magică. Nici o tehnică nu furnizează într-un mod optim o calitate a serviciilor eficientă și pe care să te poți baza. În schimb, au fost dezvoltate o varietate de tehnici, cu soluții practice care adeseori combină mai multe tehnici. Vom examina acum câteva dintre tehnicile folosite de proiectanții de sisteme pentru obținerea calității serviciilor.

Supraaprovizionarea

O soluție ușoară este aceea de a furniza ruterului suficientă capacitate, spațiu tampon și lățime de bandă încât pachetele să zboare pur și simplu. Problema cu această soluție este aceea că este costisi-

toare. Odată cu trecerea timpului și cu faptul că proiectanții au o idee mai clară despre cât de mult înseamnă suficient, această tehnică ar putea deveni chiar realistă. Până la un punct, sistemul telefonic este supraapovizionat Se întâmplă rar să ridici receptorul telefonului și să nu ai ton de formare instantaneu. Pur și simplu există atâta capacitate disponibilă încât cererea va fi întotdeauna satisfăcută.

Memorarea temporară

Fluxurile pot fi reținute în zone tampon ale receptorului înainte de a fi livrate. Aceasta nu afectează fiabilitatea sau lățimea de bandă și mărește întârzierea, dar uniformizează fluctuația. Pentru audio și video la cerere, fluctuațiile reprezintă problema principală, iar această tehnică este de mare ajutor.

Am văzut diferența dintre fluctuațiile mari și fluctuațiile mici în fig. 5-29. În fig. 5-31 vedem un flux de pachete care sunt livrate cu o fluctuație considerabilă. Pachetul 1 este transmis de către server la momentul $t = 0$ sec și ajunge la client la $t = 1$ sec. Pachetul 2 acumulează o întârziere mai mare și îi sunt necesare 2 sec pentru a ajunge. Pe măsură ce pachetele sosesc, ele sunt reținute în zona tampon pe mașina client.

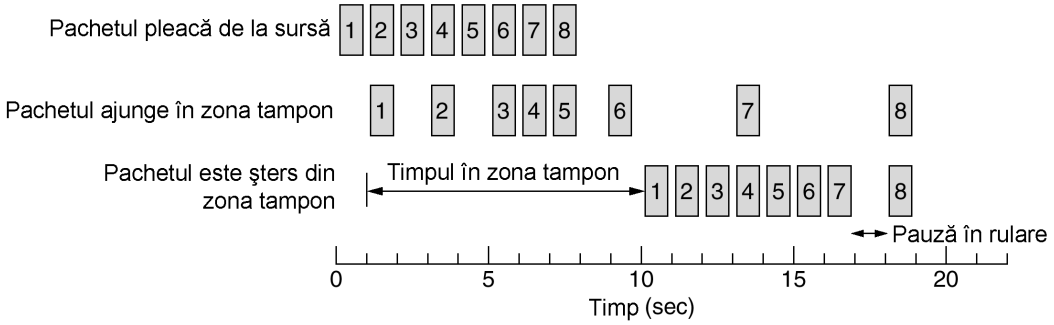

Pachetul pleacă de la sursă
1 2 3 4 5 6 7 8
Pachetul ajunge în zona tampon
1 2 3 4 5 6 7 8
Pachetul este şters din zona tampon
Timpul în zona tampon
1 2 3 4 5 6 7 8
Pauză în rulare
0 5 10 15 20
Timp (sec)

Fig. 5-31. Uniformizarea fluxului de ieșire prin memorarea temporară a pachetelor.

La $t = 10$ sec începe rularea. La acest moment, pachetele de la 1 la 6 au fost introduse în zona tampon așa că pot fi scoase de acolo la intervale egale pentru o rulare uniformă. Din păcate, pachetul 8 a avut o întârziere așa de mare încât nu este disponibil când îi vine timpul de rulare, așa că rularea trebuie întreruptă până la sosirea acestuia, creând astfel o pauză supărătoare în muzică sau în film. Această problemă poate fi alinată prin întârzierea și mai mare a momentului începerii, deși acest lucru necesită și un spațiu tampon mai mare. Sit-urile Web comerciale care conțin fluxuri audio sau video folosesc toate programe de rulare care au timpul de reținere în zona tampon de aproximativ 10 secunde până să înceapă să ruleze aplicația.

Formarea traficului

În exemplul de mai sus, sursa trimite pachetele la intervale de timp egale între ele, dar în alte cazuri ele pot fi emise neregulat, ceea ce poate duce la apariția congestiei în rețea. Neuniformitatea ieșirii este un lucru obișnuit dacă server-ul se ocupă de mai multe fluxuri la un moment dat și de asemeni permite și alte acțiuni cum ar fi derularea rapidă înainte sau înapoi, autentificarea utilizatorilor și așa mai departe. De asemenea, abordarea pe care am folosit-o aici (memorarea temporară) nu este întotdeauna posibilă, de exemplu în cazul videoconferințelor. Cu toate acestea, dacă s-ar putea face ceva pentru ca serverul (și calculatoarele gazdă în general) să transmită cu rate de transfer uniforme, calitatea serviciilor ar fi mai bună. Vom examina acum o tehnică numită **formarea traficului** (**traffic shaping**), care uniformizează traficul mai degrabă pentru server decât pentru client.

Formarea traficului se ocupă cu uniformizarea ratei medii de transmisie a datelor (atenuarea rafalelor). În contrast, protocoalele cu fereastră glisantă pe care le-am studiat anterior limitează volumul de date în tranzit la un moment dat și nu rata la care sunt transmise acestea. La momentul stabilirii unei conexiuni, utilizatorul și subrețeaua (clientul și furnizorul) stabilesc un anumit model al traficului (formă) pentru acel circuit. În unele cazuri aceasta se numește **înțelegere la nivelul serviciilor (service level agreement)**. Atât timp cât clientul își respectă partea sa de contract și trimite pachete conform înțelegerii încheiate, furnizorul promite livrarea lor în timp util. Formarea traficului reduce congestia și ajută furnizorul să-și țină promisiunea. Astfel de înțelegeri nu sunt foarte importante pentru transferul de fișiere, însă sunt deosebit de importante pentru datele în timp real, cum ar fi conexiunile audio sau video, care au cerințe stringente de calitate a serviciilor.

Pentru formarea traficului clientul spune furnizorului: „Modelul meu de transmisie arată cam așa. Poți să te descurci cu el?" Dacă furnizorul este de acord, problema care apare este cum poate spune furnizorul dacă clientul respectă înțelegerea și ce să facă dacă nu o respectă. Monitorizarea fluxului traficului se numește **supravegherea traficului (traffic policing**.) Stabilirea unei forme a traficului și urmărirea respectării ei se fac mai ușor în cazul subrețelelor bazate pe circuite virtuale decât în cazul subrețelelor bazate pe datagrame. Cu toate acestea, chiar și în cazul subrețelelor bazate pe datagrame, aceleași idei pot fi aplicate la conexiunile nivelului transport.

Algoritmul găleții găurite

Să ne imaginăm o găleată cu un mic orificiu în fundul său, așa cum este prezentată în fig. 5-32(a). Nu contează cu ce rată curge apa în găleată, fluxul de ieșire va fi la o rată constantă, ρ, dacă există apă în găleată și zero dacă găleata este goală. De asemenea, odată ce găleata s-a umplut, orice cantitate suplimentară de apă se va revărsa în afara pereților și va fi pierdută (adică nu se va regăsi în fluxul de ieșire de sub orificiu).

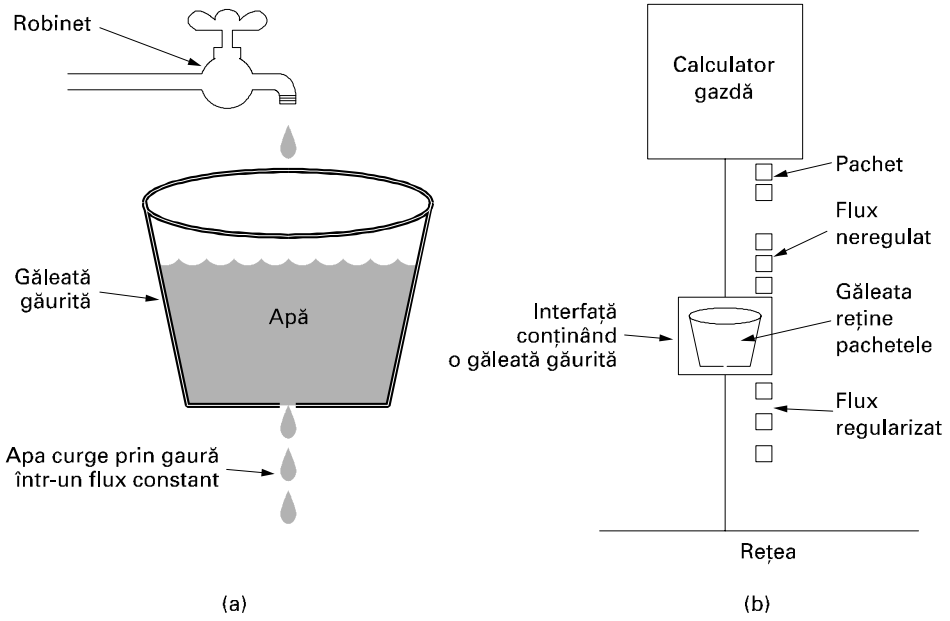


Fig. 5-32. (a) O găleată găurită umplută cu apă (b) O găleată găurită, cu pachete.

Aceeași idee poate fi aplicată și în cazul pachetelor, așa cum se arată în fig. 5-32 (b). Conceptual, fiecare calculator gazdă este conectat la rețea printr-o interfață conținând o găleată găurită, în fapt o coadă internă cu capacitate finită. Dacă un pachet sosește în coadă atunci când aceasta este plină, el este distrus. Cu alte cuvinte, dacă unul sau mai multe procese de pe calculatorul gazdă încearcă trimiterea unui pachet atunci când coada conține deja numărul maxim de pachete, pachetele noi vor fi distruse fără menajamente. Acest aranjament poate fi implementat în interfața hardware sau poate fi simulat de către sistemul de operare gazdă. A fost propus pentru prima dată de către Turner (1986) și este numit **algoritmul găleții găurite (the leaky bucket algorithm)**. De fapt nu este altceva decât un sistem de cozi cu un singur server și cu timp de servire constant.

Calculatorul gazdă poate pune în rețea câte un pachet la fiecare tact al ceasului. Și acest lucru poate fi realizat de placa de interfață sau de către sistemul de operare. Acest mecanism transformă un flux neregulat de pachete de la procesele de pe calculatorul gazdă într-un flux uniform de pachete care se depun pe rețea, netezind rafalele și reducând mult șansele de producere a congestiei.

Dacă toate pachetele au aceeași dimensiune (de exemplu celule ATM), algoritmul poate fi folosit exact așa cum a fost descris. Dacă se folosesc pachete de lungimi variabile, este adesea mai convenabil să se transmită un anumit număr de octeți la fiecare tact și nu un singur pachet. Astfel, dacă regula este 1024 octeți la fiecare tact, atunci se pot transmite un pachet de 1024 octeți, două de 512 octeți, sau patru de 256 octeți ș.a.m.d. Dacă numărul rezidual de octeți este prea mic, următorul pachet va trebui să aștepte următorul tact.

Implementarea algoritmului inițial al găleții găurite este simplă. Găleata găurită constă de fapt dintr-o coadă finită. Dacă la sosirea unui pachet este loc în coadă, el este adăugat la sfârșitul cozii, în caz contrar, este distrus. La fiecare tact se trimite un pachet din coadă (bineînțeles dacă aceasta nu este vidă).

Algoritmul găleții folosind contorizarea octeților este implementat aproximativ în aceeași manieră. La fiecare tact un contor este inițializat la n. Dacă primul pachet din coadă are mai puțini octeți decât valoarea curentă a contorului, el este transmis și contorul este decrementat cu numărul corespunzător de octeți. Mai pot fi transmise și alte pachete adiționale, atât timp cât contorul este suficient de mare. Dacă contorul scade sub lungimea primului pachet din coadă, atunci transmisia încetează până la următorul tact, când contorul este reinițializat și fluxul poate continua.

Ca un exemplu de găleată găurită, să ne imaginăm un calculator care produce date cu rata de 25 milioane octeți/sec (200 Mbps) și o rețea care funcționează la aceeași viteză. Cu toate acestea, ruterele pot să gestioneze datele la această rată doar pentru un timp foarte scurt (practic, până când li se umple spațiul tampon). Pentru intervale mai mari ele lucrează optim pentru rate care nu depășesc valoarea de 2 milioane octeți/sec. Să presupunem acum că datele vin în rafale de câte 1 milion de octeți, câte o rafală de 40 msec în fiecare secundă. Pentru a reduce rata medie la 2 MB/sec, putem folosi o găleată găurită având $\rho = 2$ MB/sec și o capacitate, C, de 1 MB. Aceasta înseamnă că rafalele de până la 1 MB pot fi gestionate fără pierderi de date și că acele rafale sunt împrăștiate de-a lungul a 500 msec, indiferent cât de repede sosesc.

În fig. 5-33(a) vedem intrarea pentru găleata găurită funcționând la 25 MB/sec pentru 40 msec. În fig. 5-33(b) vedem ieșirea curgând la o rată uniformă de 2 MB/sec pentru 500 msec.

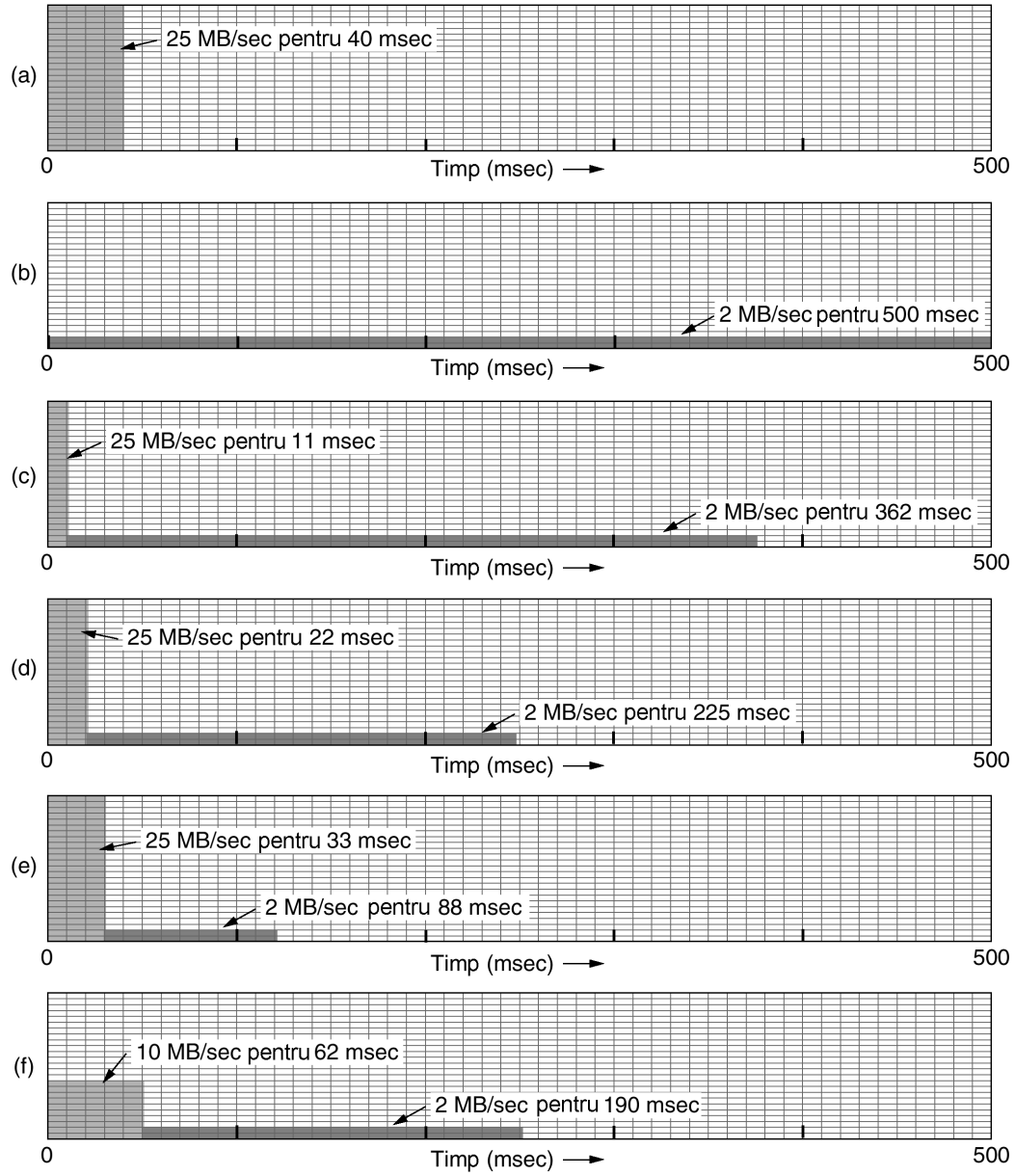


Fig. 5-33. (a) Intrarea pentru o găleată găurită. (b) Ieșirea pentru o găleată găurită. (c) - (e) Ieșirea pentru o găleată cu jeton de capacitate 250 KB, 500 KB, 750 KB. (f) Ieșirea pentru o găleată cu jeton de capacitate 500 KB care alimentează o găleată găurită de 10 MB/sec.

Algoritmul găleții cu jeton

Algoritmul găleții găurite impune un model rigid al ieșirii, din punct de vedere al ratei medii, indiferent de cum arată traficul. Pentru numeroase aplicații este mai convenabil să se permită o creștere a vitezei de ieșire la apariția unor rafale mari, astfel încât este necesar un algoritm mai flexibil, de preferat unul care nu pierde date. Un astfel de algoritm este **algoritmul găleții cu jeton (the token bucket algorithm)**. În acest algoritm, găleata găurită păstrează jetoane, generate de un ceas cu rata de un jeton la fiecare ΔT sec. În fig. 5-34(a) vedem o găleată păstrând trei jetoane și având cinci pachete care așteaptă să fie transmise. Pentru ca un pachet să fie transmis, el trebuie să captureze și să distrugă un jeton. În fig. 5-34(b) vedem că trei din cele cinci pachete au trecut mai departe, în timp ce celelalte două așteaptă să fie generate alte două jetoane.

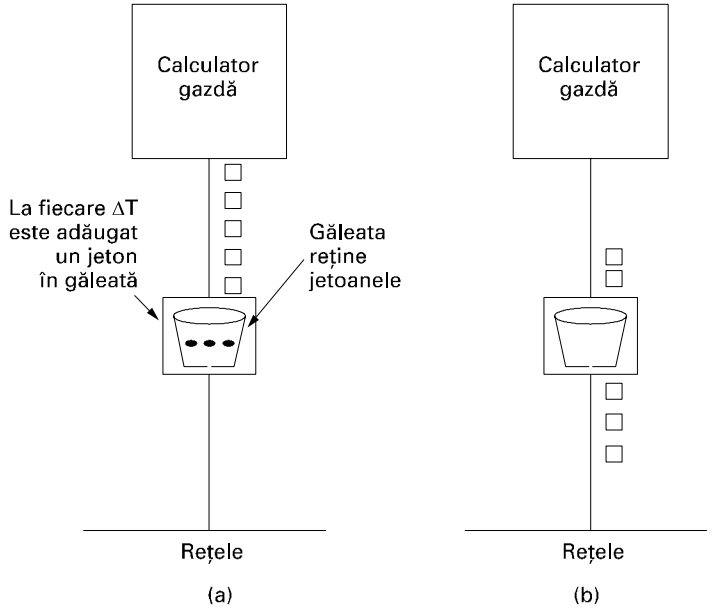


Fig. 5-34. Algoritmul găleții cu jeton. (a) Înainte. (b) După.

Algoritmul găleții cu jeton asigură o formare diferită a traficului, comparativ cu algoritmul găleții găurite. Algoritmul găleții găurite nu permite calculatoarelor gazdă inactive să acumuleze permisiunea de a trimite rafale mari ulterior. Algoritmul găleții cu jeton permite această acumulare, mergând până la dimensiunea maximă a găleții, n. Această proprietate permite ca rafale de până la n pachete să fie trimise dintr-o dată, permițând apariția unor rafale la ieșire și asigurând un răspuns mai rapid la apariția bruscă a unor rafale la intrare.

O altă diferență între cei doi algoritmi este aceea că algoritmul găleții cu jeton aruncă jetoanele (capacitatea de transmisie) la umplerea găleții, dar niciodată nu distruge pachete. Prin contrast, algoritmul găleții găurite distruge pachete la umplerea găleții.

Și aici este posibilă o variantă, în care fiecare jeton reprezintă dreptul de a trimite nu un pachet, ci k octeți. Un pachet va putea fi trimis doar dacă se dețin suficiente jetoane pentru a-i acoperi lungimea exprimată în octeți. Jetoanele fracționare sunt păstrate pentru utilizări ulterioare.

Algoritmul găleții găurite și algoritmul găleții cu jeton pot fi folosite și pentru a uniformiza traficul între rutere, la fel de bine cum pot fi folosite pentru a regla informația de ieșire a unui calculator

gazdă, ca în exemplul prezentat. Oricum, o diferență evidentă este aceea că algoritmul găleții cu jeton folosit pentru a regla ieșirea unui calculator gazdă îl poate opri pe acesta să trimită date atunci când apare vreo restricție. A spune unui ruter să oprească transmisiile în timp ce datele de intrare continuă să sosească, poate duce la pierderea de date.

Implementarea de bază a algoritmului găleții cu jeton se face printr-o variabilă care numără jetoane. Acest contor crește cu 1 la fiecare ΔT și scade cu 1 ori de câte ori este trimis un pachet. Dacă acest contor atinge valoarea zero, nu mai poate fi trimis nici un pachet. În varianta la nivel de octeți, contorul este incrementat cu k octeți la fiecare ΔT și decrementat cu lungimea pachetului trimis.

Caracteristica esențială a algoritmului găleții cu jeton este că permite rafalele, dar până la o lungime maximă controlată. Priviți de exemplu fig. 5-33(c). Aici avem o găleată cu jeton, de capacitate 250 KB. Jetoanele sosesc cu o rată care asigură o ieșire de 2 MB/sec. Presupunând că găleata este plină când apare o rafală de 1 MB, ea poate asigura scurgerea la întreaga capacitate de 25 MB/sec pentru circa 11 milisecunde. Apoi trebuie să revină la 2 MB/sec până ce este trimisă întreaga rafală.

Calculul lungimii rafalei de viteză maximă este puțin mai special. Nu este doar 1 MB împărțit la 25 MB/sec deoarece, în timp ce rafala este prelucrată, apar alte jetoane. Dacă notăm S cu lungimea rafalei în secunde, cu C capacitatea găleții în octeți, cu ρ rata de sosire a jetoanelor în octeți/secundă, cu M rata maximă de ieșire în octeți/secundă, vom vedea că o rafală de ieșire conține cel mult $C + \rho S$ octeți. De asemenea, mai știm că numărul de octeți într-o rafală de viteză maximă de S secunde este MS. De aici avem:

$$C + \rho S = MS$$

Rezolvând această ecuație obținem $S = C / (M - \rho)$. Pentru parametrii considerați $C = 250$ KB, $M = 25$ MB/sec și $\rho = 2$ MB/sec vom obține o durată a rafalei de circa 11 msec. Fig. 5-33(d) și fig. 5-33(e) prezintă găletile cu jeton de capacități 500 KB și respectiv 750 KB.

O problemă potențială a algoritmului găleții cu jeton este aceea că și el permite apariția unor rafale mari, chiar dacă durata maximă a unei rafale poate fi reglată prin selectarea atentă a lui ρ și M. De multe ori este de dorit să se reducă valoarea de vârf, dar fără a se reveni la valorile scăzute permise de algoritmul original, al găleții găurite.

O altă cale de a obține un trafic mai uniform este de a pune o găleată găurită după cea cu jeton. Rata găleții găurite va trebui să fie mai mare decât parametrul ρ al găleții cu jeton, însă mai mică decât rata maximă a rețelei. Fig. 5-25(f) ilustrează comportarea unei găleți cu jeton de 500 KB, urmată de o găleată găurită de 10 MB/sec.

Gestionarea tuturor acestor scheme poate fi puțin mai specială. În esență, rețeaua trebuie să simuleze algoritmul și să se asigure că nu se trimit mai multe pachete sau mai mulți octeți decât este permis. Totuși, aceste instrumente furnizează posibilități de a aduce traficul rețelei la forme mai ușor de administrat pentru a ajuta la îndeplinirea condițiilor necesare calității serviciilor.

Rezervarea resurselor

Un bun început pentru garantarea calității serviciilor este capabilitatea de a regla forma traficului oferit. Totuși, folosirea efectivă a acestei informații înseamnă implicit să ceri ca toate pachetele dintr-un flux să urmeze aceeași rută. Răspândirea lor aleatoare pe rutere face dificil de garantat orice. Ca o consecință, trebuie ca între sursă și destinație să se creeze ceva similar unui circuit virtual și ca toate pachetele care aparțin fluxului să urmeze această cale.

Odată ce avem o rută specifică pentru un flux, devine posibil să rezervi resurse de-a lungul acelei rute pentru a te asigura că este disponibilă capacitatea necesară. Pot fi rezervate trei tipuri diferite de resurse:

1. Lățimea de bandă
2. Zona tampon
3. Ciclurile procesorului

Prima, lățimea de bandă este cea mai evidentă. Dacă un flux cere 1 Mbps și linia pe care se iese are capacitatea de 2 Mbps, încercarea de a dirija trei fluxuri prin acea linie nu va reuși. Astfel, a rezerva lățime de bandă înseamnă să nu se supraîncarce vreo linie de ieșire.

O a doua resursă care este adeseori insuficientă este spațiul tampon. Când sosește un pachet, acesta este de obicei depozitat pe placa de rețea de către hardware. Software-ul ruterului trebuie să copieze apoi pachetul într-o zonă tampon din RAM și să adauge acest tampon în coada pentru transmisie pe linia de ieșire aleasă. Dacă nu este disponibil nici un tampon, pachetul trebuie aruncat deoarece nu există spațiu în care să poată fi pus. Pentru o bună calitate a serviciilor, unele zone tampon pot fi rezervate pentru un anumit flux în așa fel încât acel flux să nu trebuiască să concureze pentru tampoane cu alte fluxuri. Întotdeauna va exista o zonă tampon disponibilă atunci când fluxul va avea nevoie, dar până la o anumită limită.

În cele din urmă, ciclurile procesorului sunt de asemenea resurse rare. Ruterul are nevoie de timp de procesor pentru a prelucra un pachet, deci un ruter poate procesa doar un anumit număr de pachete pe secundă. Este necesar să ne asigurăm că procesorul nu este supraîncărcat pentru a asigura prelucrarea în timp util a fiecărui pachet.

La prima vedere ar putea părea că dacă, să zicem, un ruter are nevoie de 1μsec pentru a procesa un pachet, atunci el poate procesa 1 milion de pachete/sec. Această observație nu este adevărată pentru că vor exista întotdeauna perioade nefolosite datorate fluctuațiilor statistice ale încărcării. Dacă procesorul are nevoie de fiecare ciclu de ceas pentru a-și face treaba, pierderea chiar și a câtorva cicluri din cauza perioadelor de nefolosire ocazionale creează un decalaj de care nu se poate scăpa.

Oricum, chiar și cu o încărcare puțin sub capacitatea teoretică, se pot forma cozi și pot apărea întârzieri. Să considerăm o situație în care pachetele sosesc aleator, cu rată medie de sosire de λ pachete/sec. Timpul de procesor cerut de fiecare pachet este de asemenea aleator, cu o capacitate medie de procesare de μ pachete/sec. Presupunând că distribuțiile sosirii și servirii sunt distribuții Poisson, folosind teoria cozilor se poate demonstra că întârzierea medie a unui pachet, *T*, este:

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda / \mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

unde $\rho = \lambda/\mu$ este utilizarea procesorului. Primul factor, $1/\mu$, reprezintă timpul de servire în absența competiției. Al doilea factor este încetinirea cauzată de competiția cu alte fluxuri. De exemplu, dacă $\lambda = 950{,}000$ de pachete/sec și $\mu = 1{,}000{,}000$ de pachete/sec, atunci $\rho = 0.95$ și întârzierea medie a unui pachet va fi de 20 μsec în loc de 1 μsec Acest timp ia în considerare atât timpul de așteptare în coadă, cât și timpul de servire, după cum se poate vedea atunci când încărcarea este foarte mică ($\lambda/\mu \approx 0$). Dacă presupunem că pe traseul fluxului există 30 de rutere, numai datorită întârzierilor în cozi vor rezulta 600μsec de întârziere.

Controlul accesului

Acum suntem în punctul în care traficul dintr-un flux oarecare este bine format și poate să urmeze o singură rută în care capacitatea poate fi rezervată în avans pe ruterele ce se găsesc de-a lungul căii. Când un asemenea flux este oferit unui ruter, acesta trebuie să decidă, pe baza capacității sale și a numărului de angajamente pe care le-a făcut deja cu alte fluxuri, dacă să primească sau să respingă fluxul.

Decizia de a accepta sau de a respinge fluxul nu este o simplă chestiune de comparație între (lățime de bandă, zone tampon, cicluri) cerute de către flux și capacitatea în exces a ruterului în aceste trei dimensiuni. Este puțin mai complicat decât atât. În primul rând, deși unele aplicații și-ar putea cunoaște necesitățile de lățime de bandă, puține știu despre zonele tampon sau despre ciclurile de procesor, deci, la nivel minim, este necesar un alt mod de descriere a fluxurilor. Apoi, unele aplicații sunt de departe mai tolerante la ratarea ocazională a unui termen limită. În cele din urmă, unele aplicații ar putea fi dornice să negocieze în legătură cu parametrii fluxului, iar altele nu. De exemplu, un server video care rulează în mod normal la 30 de cadre/sec ar putea fi dispus să coboare la 25 de cadre /sec dacă nu există suficientă lățime de bandă liberă pentru a suporta 30 de cadre/sec. În mod similar pot fi ajustați numărul de pixeli pe cadru, lățimea de bandă audio și alte proprietăți.

Deoarece în negocierea fluxului pot fi implicate mai multe părți (emițătorul, receptorul și toate ruterele aflate de-a lungul căii dintre aceștia), fluxurile trebuie descrise exact în termenii parametrilor specifici ce pot fi negociați. Un set de astfel de parametri se numește **specificarea fluxului (flow specification)**. Tipic, emițătorul (adică serverul video) produce o specificație a fluxului propunând parametrii pe care ar vrea să îi folosească. Pe măsură ce specificația se propagă de-a lungul căii, fiecare ruter o examinează și îi modifică parametrii după cum are nevoie. Modificările pot doar să reducă fluxul, nu să îl și crească (de exemplu o rată mai mică a datelor, nu mai mare). Când ajunge la capătul celălalt, parametrii pot fi stabiliți.

Ca un exemplu de ce poate exista într-o specificație a fluxului, să considerăm exemplul din fig. 5-35, care se bazează pe RFC-urile 2210 și 2211. Are cinci parametri, dintre care primul, *rata găleții cu jeton*, reprezintă numărul de octeți pe secundă care sunt puși în găleată. Aceasta este rata maximă suportată la care poate transmite emițătorul, mediată de-a lungul unui interval mare de timp.

Parametru	Unitate de măsură
Rata găleții cu jeton	Octeți/sec
Dimensiunea găleții cu jeton	Octeți
Rata de vârf a datelor	Octeți/sec
Dimensiunea minimă a pachetului	Octeți
Dimensiunea maximă a pachetului	Octeți

Fig. 5-35. Un exemplu de specificație a fluxului

Al doilea parametru este dimensiunea găleții în octeți. Dacă, de exemplu, *rata găleții cu jeton* este de 1 Mbps și *dimensiunea găleții cu jeton* este de 500 KB, găleata se poate umplută continuu timp de 4 sec până să fie plină (în absența oricărei transmisii). Orice jeton trimis după aceasta este pierdut.

Al treilea parametru, *Rata de vârf a datelor* este rata de transmisie maximă tolerată, chiar și pentru intervale scurte de timp. Emițătorul nu trebuie să depășească niciodată această valoare.

Ultimii doi parametri specifică dimensiunile minimă și maximă ale pachetului, incluzând antetele nivelelor transport și rețea (de exemplu TCP și IP). Dimensiunea minimă este importantă deoarece durata procesării fiecărui pachet este fixată, indiferent cât este de mic pachetul. Un ruter poate fi pregătit să se ocupe de 10,000 de pachete/sec de câte 1 KB fiecare, dar nepregătit să se ocupe de 100000 de pachete/sec de câte 50 de octeți fiecare, chiar dacă acestea reprezintă mai puține date. Dimensiu-

nea maximă a pachetului este importantă datorită limitărilor interne ale rețelei, care nu pot fi depășite. De exemplu, dacă o parte a căii trece prin Ethernet, dimensiunea maximă a pachetului va fi restricționată la nu mai mult de 1500 de octeți, indiferent de cât poate să suporte restul rețelei.

O întrebare interesantă este cum transformă un ruter o specificație a fluxului într-un set de rezervări specifice de resurse. Această mapare este implementată specific și nu este standardizată. Să presupunem că un ruter poate procesa 100000 pachete/sec. Dacă îi este oferit un flux de 1MB/sec cu dimensiunile minimă și maximă a pachetului de 512 octeți, ruterul poate calcula că ar putea lua 2048 de pachete/sec din acel flux. În acest caz, el trebuie să rezerve 2% din procesor pentru acel flux, preferabil mai mult pentru a evita întârzierile cauzate de așteptarea în coadă. Dacă politica unui ruter este de a nu aloca niciodată mai mult de 50 % din procesor (ceea ce implică o întârziere dublă) și este deja solicitat 49%, atunci acest flux trebuie respins. Calcule asemănătoare sunt necesare și pentru celelalte resurse.

Cu cât este mai exigentă specificația, cu atât ea este mai folositoare ruterelor. Dacă o specificație de flux spune că are nevoie de *o rată a găleții cu jeton* de 5MB/sec dar pachetele pot varia de la 50 de octeți la 1500 de octeți, atunci rata pachetelor va varia de la circa 3500 de pachete/sec la 105000 de pachete/sec. Ruterul s-ar putea panica la acest număr și ar putea respinge fluxul, deși la o dimensiune minimă a pachetului de 1000 de octeți, fluxul de 5MB/sec ar fi fost acceptat.

Dirijarea proporțională

Cei mai mulți algoritmi de dirijare încearcă să găsească cea mai bună cale pentru fiecare destinație și îndreaptă tot traficul spre acea destinație pe cea mai bună cale. O abordare diferită care a fost propusă pentru a furniza o mai bună calitate a serviciilor este de a împărți traficul pentru fiecare destinație pe mai multe căi. Din moment ce ruterele nu au în general o imagine de ansamblu completă asupra traficului din rețea, singura posibilitate reală de a împărți traficul pe mai multe rute este de a folosi informațiile disponibile local. O metodă simplă este de a împărți traficul în mod egal sau proporțional cu capacitatea legăturilor de ieșire. Totuși, sunt disponibili și algoritmi mai sofisticați (Nelakuditi și Zhang, 2002).

Planificarea pachetelor

Dacă un ruter se ocupă de mai multe fluxuri, există pericolul ca un flux să ia prea mult din capacitate, înghețând toate celelalte fluxuri. Procesare pachetelor în ordinea sosirii lor înseamnă că un emițător agresiv poate acapara cea mai mare parte din capacitatea ruterelor pe care le traversează pachetele sale, micșorând calitatea serviciilor pentru celelalte. Pentru a împiedica asemenea tentative au fost inventați diverși algoritmi de planificare a pachetelor (Bhatti și Crowcroft, 2000).

Unul dintre primii a fost algoritmul **așteptare echitabilă (fair queuing)** (Nagle, 987). Esența algoritmului este că ruterele au cozi separate pentru fiecare linie de ieșire, câte una pentru fiecare flux. Când o linie se eliberează, ruterul scanează cozile după metoda round robin, luând primul pachet din coada următoare. În acest fel, cu n calculatoare gazdă care concurează pentru o anumită linie de ieșire, fiecare gazdă reușește să expedieze un pachet din fiecare n. Trimiterea mai multor pachete nu va îmbunătăți rata.

Deși este un început, algoritmul are o problemă: dă mai multă lățime de bandă gazdelor care folosesc pachete mari decât celor care folosesc pachete mici. Demers ș.a. (1990) a sugerat o îmbunătățire în care parcurgerea round robin este realizată astfel încât să simuleze un round-robin octet-cu-octet în locul unui round robin pachet-cu-pachet. În concluzie, el scanează cozile în mod repetat, octet cu octet, până găsește momentul la care fiecare pachet va fi terminat. Apoi pachetele sunt sortate în ordine terminării lor și trimise în acea ordine. Algoritmul este ilustrat în fig. 5-36.

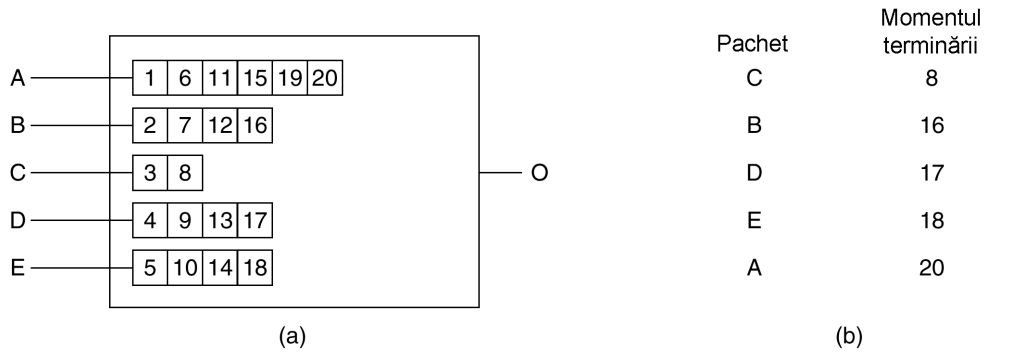


Fig. 5-36. (a) Un ruter cu 5 pachete așteptând pentru linia *O*. (b) Momentul terminării pentru cele 5 pachete.

În fig. 5-36(a) observăm pachete cu lungimi între 2 și 6 octeți. La momentul (virtual) 1 este trimis primul octet al pachetului de pe linia A. Apoi urmează primul octet al pachetului de pe linia B și așa mai departe. Primul pachet care este terminat este C, după opt perioade. Ordinea sortată este dată în fig. 5-36(b). În absența unor noi sosiri, pachetele vor fi trimise în ordinea listată, de la C la A.

O problemă a acestui algoritm este aceea că acordă tuturor calculatoarelor gazdă aceeași prioritate. În multe situații, este de dorit să se acorde mai multă lățime de bandă serverelor video decât serverelor de fișiere normale în așa fel încât să poată transmită doi sau mai mulți octeți pe perioadă. Acest algoritm modificat se numește **așteptare echitabilă ponderată (weighted fair queueing)** și este foarte folosit. Uneori ponderea este egală cu numărul de fluxuri care ies dintr-un calculator, astfel încât toate procesele primesc lățime de bandă egală. O implementare eficientă a acestui algoritm este discutată în (Shreedhar și Varghese, 1995). Din ce în ce mai mult, retransmiterea efectivă a pachetelor printr-un ruter sau comutator se face prin hardware (Elhananz et al., 2001).

5.4.3 Servicii integrate

Între 1995 și 1997, IETF a depus mult efort pentru a inventa o arhitectură pentru fluxurile de tip multimedia. Această muncă s-a concretizat în peste doua duzini de RFC-uri, începând cu RFC-urile 2205-2210. Numele generic al acestui rezultat este **algoritmi bazați pe flux (flow-based algorithms)** sau **servicii integrate (integrated services)**. A fost gândit atât pentru aplicații cu trimitere unică (eng.: unicast) cât și pentru cele cu trimitere multiplă (eng.: multicast). Un exemplu pentru cele dintâi este cazul unui singur utilizator rulând o secvență video de pe un sit de știri. Un exemplu pentru cel din urmă este o colecție de stații de televiziune prin cablu difuzându-și programele ca șiruri de pachete IP mai multor receptori din diverse locații. În cele ce urmează ne vom concentra pe multicast, având în vedere că unicastul este un caz particular al multicast-ului.

În numeroase aplicații multicast, grupurile își pot schimba dinamic componența, de exemplu ca participanții la o videoconferință care se plictisesc și comută pe un canal cu un serial ușurel sau pe canalul de crochet. În aceste condiții, abordarea în care emițătorii rezervă lățime de bandă în avans nu mai funcționează corespunzător, deoarece va cere fiecărui emițător să urmărească toate intrările și ieșirile celor din audiența sa. Pentru un sistem destinat transmisiilor televiziune cu milioane de abonați, nu va merge deloc.

RSVP - Protocol de rezervare a resurselor

Principalul protocol IETF pentru arhitectura serviciilor integrate este **RSVP (Resource reSerVation Protocol)**. El este descris în RFC 2205 și altele. Acest protocol este folosit pentru a face rezervări; pentru transmisia datelor sunt folosite alte protocoale. RSVP permite emițătorilor multipli să transmită spre grupuri multiple de receptori, permite fiecărui receptor să schimbe canalul la alegere și optimizează lățimea de bandă folosită, eliminând în același timp congestia.

În forma sa cea mai simplă, protocolul folosește dirijarea multicast cu arbori de acoperire, așa cum s-a discutat anterior. Fiecare grup are asociată o adresă de grup. Pentru a trimite unui grup, emițătorul pune adresa grupului în pachetele pe care le trimite. În continuare, algoritmul standard de dirijare multicast va construi un arbore de acoperire care acoperă toți membrii grupului. Algoritmul de dirijare nu este parte a RSVP. Singura diferență față de multicast-ul normal este o mică informație suplimentară distribuită periodic grupului pentru a spune ruterelor de-a lungul arborelui să mențină anumite structuri de date.

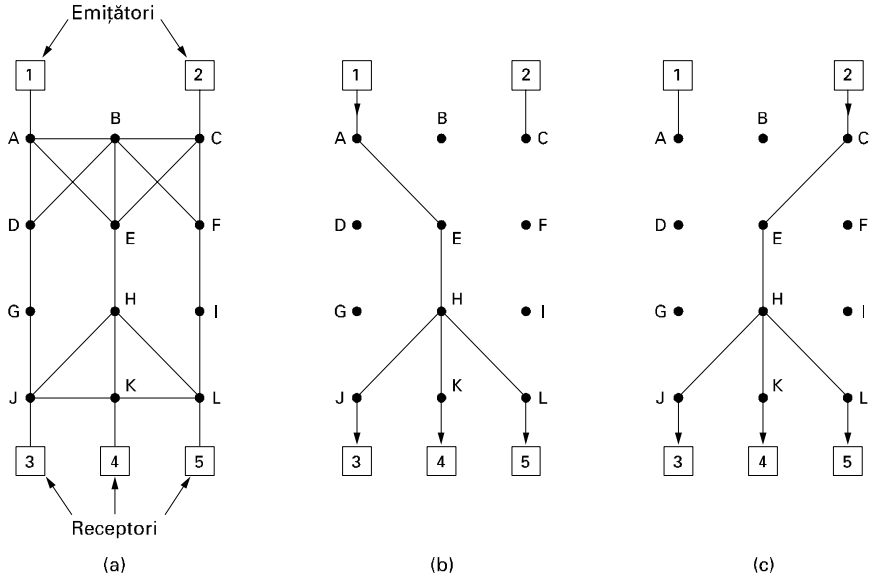


Fig. 5-37. (a) O rețea. (b) Arborele de acoperire multicast pentru calculatorul 1. (c) Arborele de acoperire multicast pentru calculatorul 2.

Ca exemplu, să considerăm rețeaua din fig. 5-37(a). Calculatoarele gazdă 1 și 2 sunt emițători multicast, iar calculatoarele gazdă 3, 4 și 5 sunt receptori multicast. În acest exemplu emițătorii și receptorii sunt disjuncți, dar în general cele două mulțimi se pot suprapune. Arborii multicast pentru mașinile 1 și 2 sunt prezentați în fig. 5-37(b), respectiv fig. 5-37(c).

Pentru a avea o recepție mai bună și pentru a elimina congestia, fiecare dintre receptorii dintr-un grup poate să trimită un mesaj de rezervare în sus pe arbore spre emițător. Mesajul este propagat folosind algoritmul căii inverse discutat anterior. La fiecare salt, ruterul notează rezervarea și rezervă lățimea de bandă necesară. Dacă lățimea de bandă este insuficientă, se raportează eroare. Atunci când mesajul ajunge la sursă, lățimea de bandă a fost rezervată pe tot drumul de la emițător spre receptorul care a făcut rezervarea de-a lungul arborelui de acoperire.

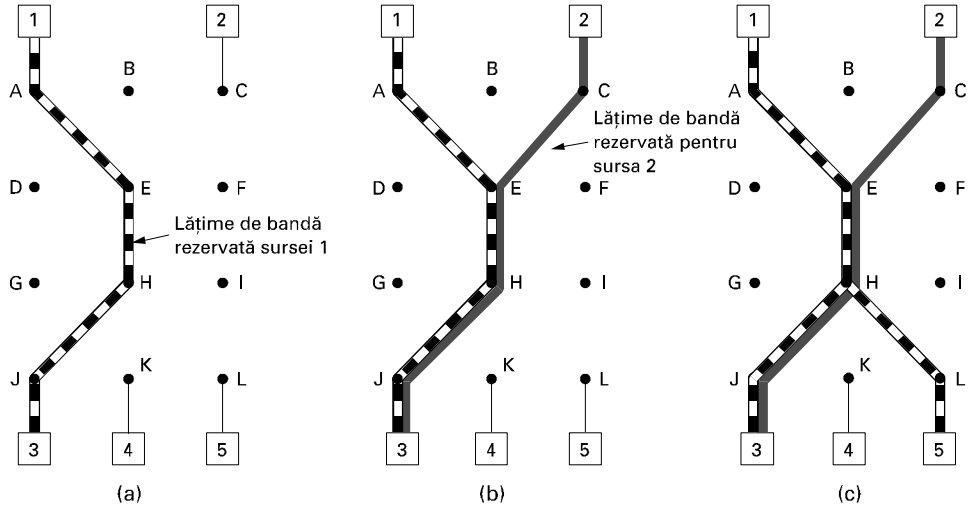


Fig. 5-38. (a) Calculatorul gazdă 3 cere un canal către calculatorul gazdă 1. (b) Calculatorul gazdă 3 cere un al doilea canal către calculatorul gazdă 2. (c) Calculatorul gazdă 5 cere un canal către calculatorul gazdă 1.

Un exemplu de astfel de rezervare este prezentat în fig. 5-38(a). Aici calculatorul gazdă 3 a cerut un canal către calculatorul gazdă 1. Odată ce acesta a fost stabilit, pachetele pot ajunge de la 1 la 3 fără congestie. Să vedem acum ce se întâmplă dacă gazda 3 rezervă și un canal către celălalt emițător, calculatorul gazdă 2, astfel încât utilizatorul să poată urmări două posturi de televiziune simultan. Se rezervă o a doua cale, așa cum se ilustrează în fig. 5-38(b). Observați că sunt necesare două canale distincte de la calculatorul gazdă 3 către ruterul *E*, deoarece se transmit două fluxuri independente.

În fine, în fig. 5-38 (c) calculatorul gazdă 5 decide să urmărească programul de televiziune transmis de 1 și face de asemenea o rezervare. Pentru început, se rezervă lățime de bandă spre ruterul *H*. Acest ruter observă că are deja o rezervare către 1, astfel încât dacă lățimea de bandă necesară a fost rezervată, nu mai trebuie să rezerve nimic. Se poate întâmpla ca 3 și 5 să ceară lățimi de bandă diferite (de exemplu, 3 are un sistem de televiziune alb-negru, astfel încât nu are nevoie de informația de culoare), caz în care capacitatea rezervată trebuie să fie suficientă pentru a satisface cererea cea mai mare.

La realizarea unei rezervări, un receptor poate (opțional) specifica una sau mai multe surse de la care vrea să recepționeze. De asemenea, el mai poate specifica dacă aceste alegeri sunt fixe pe durata rezervării sau dacă receptorul dorește să-și păstreze opțiunea de a modifica sursele ulterior. Ruterele folosesc această informație pentru a optimiza planificarea lățimii de bandă. În particular, doi receptori pot să partajeze o cale doar dacă ambii stabilesc să nu modifice sursele ulterior.

Motivul acestei strategii în cazul dinamic sută la sută este că rezervarea lățimii de bandă este decuplată de alegerea sursei. Odată ce un receptor a rezervat lățime de bandă, el poate comuta către altă sursă și să păstreze porțiunea din calea existentă care este comună cu calea către noua sursă. Când calculatorul 2 transmite mai multe fluxuri video, de exemplu, calculatorul 3 poate comuta între ele după dorință, fără a-și schimba rezervarea: ruterelor nu le pasă la ce program se uită utilizatorul.

5.4.4 Servicii diferențiate

Algoritmii bazați pe flux au potențialul de a oferi o bună calitate a serviciilor unuia sau mai multor fluxuri deoarece acestea își rezervă resursele necesare de-a lungul rutei. Totuși ei au și un dezavantaj: au nevoie să stabilească în avans caracteristicile fiecărui flux, ceea ce nu este optim în cazul în care există milioane de fluxuri. De asemenea, ei memorează caracteristicile fiecărui flux ca date interne ale ruterului, ceea ce le face vulnerabile în cazul căderii ruterului. În fine, modificările necesare codului ruterului sunt substanțiale și implică schimburi complexe de informații între rutere pentru stabilirea fluxurilor. Ca o consecință, există foarte puține implementări ale RSVP-ului sau a ceva asemănător.

Din aceste motive, IETF a propus și o abordare mai simplă a calității serviciilor, care poate fi implementată local în fiecare ruter fără inițializări prealabile și fără a implica întreaga cale. Această abordare este cunoscută sub numele de calitatea serviciilor **orientate pe clase (class-based)** (opusă abordării bazate pe flux). IETF a standardizat o arhitectură numită **servicii diferențiate** (**differentiated services**), care este descrisă în RFC-urile 2474, 2475 și altele. O vom descrie în continuare.

Serviciile diferențiate (DS) pot fi oferite de către un set de rutere care formează un domeniu administrativ (de exemplu un ISP sau telco). Administrația definește un set de clase de servicii cu regulile de rutare corespunzătoare. Dacă un client se înscrie pentru DS, pachetele clientului care intră în domeniu pot avea un câmp *Type of Service (Tipul serviciului)*, cu servicii mai bine furnizate unor clase (de exemplu serviciu premium) decât altora. Traficului dintr-o clasă i se poate cere să se conformeze unei anumite forme, cum ar fi găleata găurită cu o anumită rată decurgere. Un operator cu fler ar putea să ceară mai mult pentru fiecare pachet premium transportat sau ar putea să permită maximum N pachete premium pe lună pentru o taxă lunară suplimentară fixă. Observați că această schemă nu presupune inițializarea în avans, nici rezervarea resurselor și nici negocierea capăt-la-capăt pentru fiecare flux, care consumă timp, ca în cazul serviciilor integrate. Aceasta face ca serviciile diferențiate să fie relativ ușor de implementat.

Serviciile bazate pe clase se întâlnesc și în alte domenii. De exemplu, companiile care livrează pachete oferă de obicei servicii de tip "peste noapte", "în două zile" sau "în trei zile". Companiile aeriene oferă clasa întâi, clasa de afaceri și clasa a doua. Trenurile pe distanțe lungi oferă adeseori multiple clase de servicii. Chiar și metroul din Paris are două tipuri de servicii. Pentru pachete, clasele pot să difere, printre altele, prin valorile întârzierii, fluctuației și a probabilității pachetului de a fi aruncat la apariția unei congestii.

Pentru a evidenția diferența dintre calitatea serviciilor bazate pe flux și calitatea serviciilor bazate pe clase, să considerăm un exemplu: telefonia Internet. În cazul organizării pe flux, fiecare apel telefonic are propriile resurse și garanții. În cazul organizării pe clase, toate apelurile telefonice beneficiază de resursele rezervate pentru clasa telefoniei. Aceste resurse nu pot fi preluate de pachete din clasa transferului de fișiere sau din alte clase, dar nici un apel telefonic nu beneficiază de resurse particulare, rezervate doar pentru acesta.

Retransmitere expeditivă (Expedited Forwarding)

Alegerea claselor de servicii este la dispoziția fiecărui operator, dar, având în vedere că pachetele sunt adesea expediate între subrețele administrate de operatori diferiți, IETF lucrează la definirea unor clase de servicii independente de rețea. Cea mai simplă clasă este **retransmiterea expeditivă (expedited forwarding)**, deci vom începe cu aceasta. Descrierea ei este dată în RFC 3246.

Ideea care stă la baza retransmiterii expeditive este foarte simplă. Sunt disponibile două clase de servicii: normale și rapide (expeditive). Marea majoritate a traficului se presupune că este de tip normal, dar o mică parte se face și expeditiv. Pachetele din această clasă ar trebui să poată traversa subrețeaua ca și cum nu ar mai exista și alte pachete. O reprezentare simbolică a acestui sistem "bi-canal" este dată în fig. 5-39. Observați că există doar o singură linie fizică. Cele două canale logice din figură reprezintă o cale de a rezerva lățime de bandă și nu o a doua linie fizică.

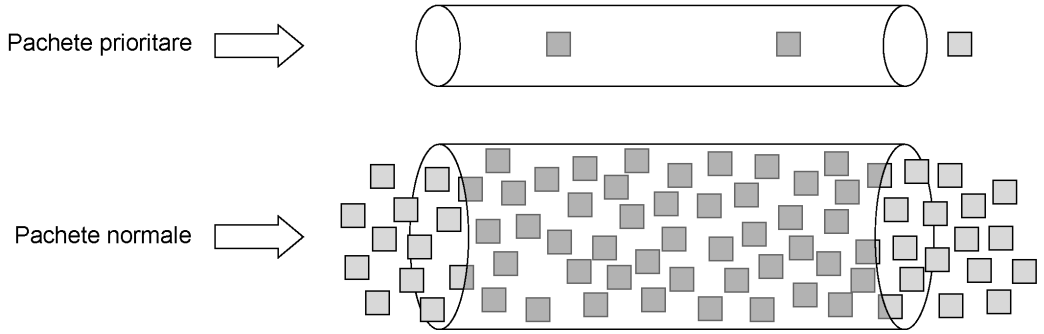


Fig. 5-39. Pachetele prioritare trec printr-o rețea cu trafic redus.

O modalitate de a implementa această strategie este de a programa ruterele astfel încât să aibă două cozi de așteptare pentru fiecare linie de ieșire, una pentru pachete prioritare și cealaltă pentru pachete normale. Când sosește un pachet este introdus în coada corespunzătoare. Planificarea pachetelor ar trebui să utilizeze ceva de genul așteptării echitabile ponderate. De exemplu, dacă 10% din trafic este de tip expeditiv și 90% de tip normal, atunci 20% din lățimea de bandă ar putea fi repartizată traficului expeditiv, iar restul traficului normal. În acest fel traficul expeditiv ar avea de două ori mai multă lățime de bandă decât îi este necesar pentru a asigura întârziere mică. Această repartizare poate fi obținută transmițând câte un pachet prioritar la fiecare patru pachete normale (presupunând că distribuția dimensiunilor pachetelor ambelor clase este similară). În acest fel se speră că pachetele prioritare nu văd subrețeaua ca fiind aglomerată, chiar dacă există o încărcare substanțială.

Rutare garantată

O metodă mai elaborată de a administra clasele de servicii este **rutarea garantată (assured forwarding)**. Descrierea acesteia se găsește în RFC 2597. Ea precizează că ar trebui să existe patru clase de priorități, fiecare cu propriile resurse. În plus definește trei probabilități de aruncare a pachetelor care sunt supuse congestiei: mică, medie și mare. Luate împreună, aceste două criterii definesc 12 clase de servicii.

Fig. 5-40 prezintă o modalitate în care pachetele ar putea fi procesate în cazul rutării garantate. Primul pas este acela de a repartiza pachetele într-una din cele patru clase de priorități. Acest pas se poate face pe calculatorul emițător (după cum se vede și din figură) sau în primul ruter. Avantajul realizării clasificării pe calculatorul gazdă care realizează transmisia este acela că are la dispoziție mai multe informații despre fiecare pachet și direcția în care se îndreaptă.

Al doilea pas este de a marca pachetele în conformitate cu clasa din care fac parte. Pentru aceasta este necesar un câmp antet. Din fericire în antetul IP este disponibil un câmp pe 8 biți numit *Tipul serviciului*, după cum vom vedea în continuare. RFC 2597 specifică faptul că șase dintre acești biți se vor folosi pentru clasa de serviciu, lăsând loc pentru codificarea claselor istorice și a celor viitoare.

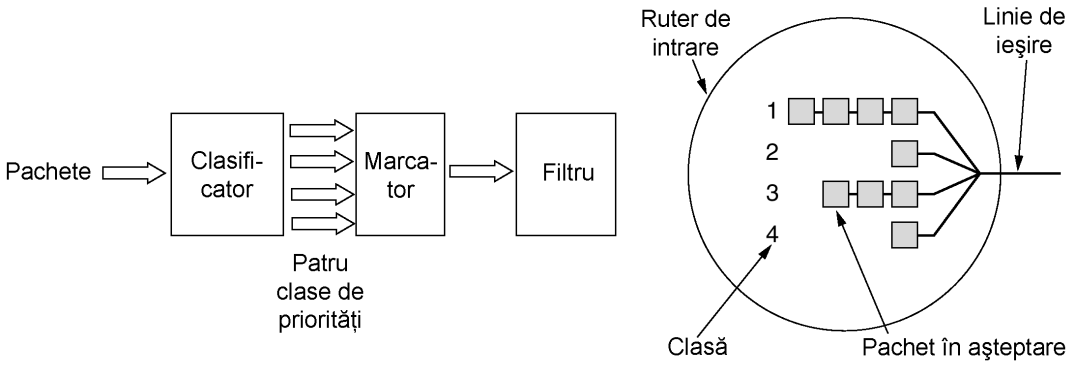


Fig. 5-40. O posibilă implementare a fluxului de date pentru rutarea garantată.

Pasul al treilea presupune trecerea pachetelor printr-un filtru (eng.: shaper/dropper filter) care ar putea întârzia sau arunca o parte dintre ele pentru a forma cele patru fluxuri într-o manieră acceptabilă, folosind de exemplu algoritmul găleții găurite sau al găleții cu jeton. Dacă sunt prea multe pachete, unele dintre ele ar putea fi aruncate în această etapă, în funcție de categoria în care au fost plasate de filtru. Sunt posibile și scheme mai elaborate care implică măsurători și reacții inverse

În acest exemplu, pașii specificați sunt efectuați pe calculatorul emițător, deci fluxul rezultat este trimis primului ruter. Merită observat faptul că acești pași pot fi efectuați de un software de rețea specializat sau chiar de către sistemul de operare, pentru a evita modificarea aplicației existente.

5.4.5 Comutarea etichetelor și MPLS

În timp ce IETF lucra la serviciile integrate și diferențiate, unii dintre vânzătorii de rutere căutau metode mai bune de rutare. Munca lor s-a axat pe adăugarea unei etichete în fața fiecărui pachet și pe efectuarea rutării mai degrabă pe baza acestei etichete decât a adresei destinație. Utilizarea etichetei ca index într-o tabelă internă a ruterului face ca găsirea liniei de ieșire corecte să se transforme într-o simplă căutare în tabel. Folosind această metodă, rutarea se poate face foarte repede iar resursele necesare pot fi rezervate pe traseu.

Bineînțeles, etichetarea fluxurilor în acest fel se apropie foarte mult de circuitele virtuale. X.25, ATM, releu de cadre (eng.: frame-relay) și toate celelalte rețele cu subrețele cu circuite virtuale pun și ele câte o etichetă (identificatorul circuitului virtual) în fiecare pachet, o caută într-o tabelă și rutarea se face pe baza intrării din tabelă. În ciuda faptului că mulți dintre membrii comunității Internet au o puternică antipatie față de rutarea orientată pe conexiune, se pare că nu se renunță la idee, de data aceasta pentru a furniza o rutare rapidă și calitatea serviciilor. Totuși există diferențe fundamentale între modul în care se ocupă Internetul de construcția rutelor și modul în care o fac rețelele orientate pe conexiune, deci în mod sigur tehnica nu este tradițională comutare de circuite.

"Noua" idee de comutare este cunoscută sub mai multe denumiri, inclusiv **comutarea etichetelor** (eng.: **label switching**) și **comutarea marcajelor** (eng.: **tag switching**). În cele din urmă, IETF a început să standardizeze ideea sub numele de **multiprotocol de comutare a etichetelor (MPLS – MultiProtocol Label Switching)**. În continuare o vom numi MPLS. Aceasta este descrisă în RFC 3031 și în multe alte RFC-uri.

Pe de altă parte, unii fac diferența între *rutare* și *comutare*. Rutarea este procesul de căutare a adresei destinație în tabel pentru a găsi unde trebuie trimis. În schimb, comutarea folosește o eti-

chetă luată dintr-un pachet ca index într-o tabelă de rutare. Aceste definiții sunt totuși departe de a fi universale.

Prima problemă care apare este unde se pune eticheta. Din moment ce pachetele IP nu au fost proiectate pentru circuite virtuale, în cadrul antetelor IP nu există nici un câmp disponibil pentru numerele circuitelor virtuale. Din acest motiv, în fața antetului IP a trebuit adăugat un nou antet MPLS. Pe o linie ruter-la-ruter folosind PPP ca protocol de încadrare, formatul cadrului, inclusiv antetele PPP, MPLS, IP și TCP, arată ca în fig. 5-41. Într-un fel, MPLS este nivelul 2.5.

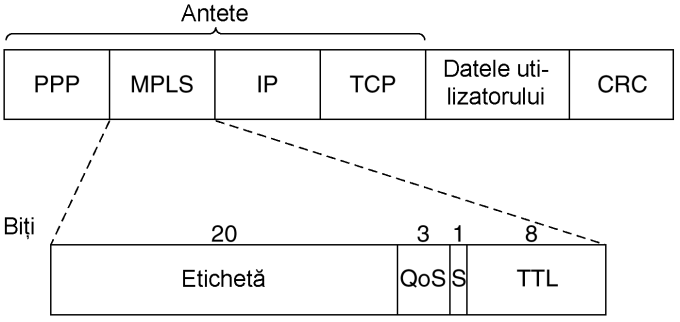


Fig. 5-41. Transmiterea unui segment TCP utilizând IP, MPLS și PPP.

Antetul generic MPLS are 4 câmpuri, dintre care cel mai important este câmpul *Etichetă (Label)* care deține indexul. Câmpul *QoS (Quality of Service; rom.: Calitatea serviciilor)* precizează clasa de servicii. Câmpul *S* se referă la memorarea mai multor etichete în rețelele ierarhice (care vor fi discutate mai târziu). Dacă ajunge la valoarea 0, pachetul este aruncat. Această facilitate previne buclarea infinită în cazul instabilității rutării.

Deoarece antetele MPLS nu fac parte din pachetul format la nivelul rețea și nici din cadrele nivelului legătură de date, MPLS reprezintă o extindere a acestor două nivele. Printre altele aceasta înseamnă că este posibilă construirea de comutatoare MPLS care să poată ruta atât pachete IP cât și celule ATM, după necesități. Această caracteristică a dus la apariția termenului "multiprotocol" din denumirea de MPLS.

Când un pachet îmbunătățit MPLS (sau o celulă) ajunge la un ruter ce suportă MPLS, eticheta este folosită ca index într-un tabel pentru a determina linia de ieșire ce va fi folosită precum și noua etichetă. Această schimbare de etichete se folosește în toate subrețelele cu circuite virtuale deoarece etichetele au doar semnificație locală și două rutere diferite pot trimite pachete care nu au legătură unul cu altul dar au aceeași etichetă unui alt ruter pentru a fi transmise pe aceeași linie de ieșire. Pentru a putea fi diferențiate la capătul celălalt, etichetele trebuie să fie remapate la fiecare salt. Am văzut cum funcționează acest mecanism în fig. 5-3. MPLS folosește aceeași tehnică.

O diferență față de circuitele virtuale tradiționale este nivelul de agregare. În mod sigur este posibil ca fiecare flux să aibă propriul set de etichete în cadrul subrețelei. Totuși se obișnuiește mai mult ca ruterele să grupeze mai multe fluxuri care au ca destinație un anumit ruter sau LAN și să folosească pentru ele o singură etichetă. Despre fluxurile care sunt grupate sub o aceeași etichetă se spune că aparțin aceleiași **Clase echivalente de rutare (FEC – Forwarding Equivalence Class)**. Această clasă acoperă nu numai pachetele care pleacă, dar și clasele lor de servicii (în sensul claselor diferențiate de servicii) deoarece toate pachetele pe care le conțin sunt tratate la fel din motive de expediere.

La rutarea tradițională de tip circuit virtual nu este posibil să se grupeze mai multe căi cu destinații diferite pe același identificator de circuit virtual, deoarece la destinația finală nu ar mai fi posibil

să se facă distincție între ele. Cu MPLS, pachetele conțin, pe lângă etichetă, și adresa lor finală de destinație, în așa fel încât, la capătul rutei etichetate, antetul poate fi eliminat și rutarea poate continua în mod obișnuit, folosind adresa de destinație a nivelului rețea.

O diferență majoră dintre MPLS și circuitele virtuale convenționale este modul în care este construită tabela de rutare. În rețelele cu circuite virtuale tradiționale, când un utilizator dorește să stabilească o conexiune, este lansat în rețea un pachet de inițializare pentru a crea calea și intrările tabelelor de rutare. MPLS nu funcționează astfel deoarece nu există fază de inițializare pentru fiecare conexiune (deoarece ar stopa prea mult din software-ul existent în Internet).

În schimb există două modalități în care pot fi create intrările tabelelor de rutare. În abordarea **bazată pe date**, **(data-driven)** atunci când un pachet ajunge la primul ruter, acesta contactează ruterul din aval, la care trebuie să ajungă pachetul, și îi cere să genereze o etichetă pentru flux. Această metodă se aplică recursiv. Practic, aceasta este crearea circuitelor virtuale la cerere.

Protocoalele care realizează această răspândire au foarte mare grijă să evite buclele. Ele folosesc adeseori tehnica numită a **firelor colorate (colored threads)**. Propagarea înapoi a unei FEC poate fi comparată cu tragerea unui fir de o singură culoare înapoi în subrețea. Dacă un ruter întâlnește o culoare pe care o are deja, el știe că există o buclă și ia măsuri pentru remedierea situației. Abordarea bazată pe date este folosită în rețelele unde sub nivelul transport se găsește ATM-ul (ca în cazul sistemului telefonic).

Cealaltă modalitate, folosită în rețele care nu sunt bazate pe ATM, este abordarea bazată pe control (eng.: **control-driven**). Ea are mai multe variante, dintre care una funcționează în felul următor. Când este pornit un ruter, verifică pentru ce rute el reprezintă destinația finală (de exemplu ce calculatoare gazdă sunt în rețeaua locală). Apoi creează una sau mai multe clase echivalente pentru acestea, alocă pentru fiecare câte o etichetă și trimite etichetele vecinilor săi. Aceștia, la rândul lor, introduc etichetele în tabelele de rutare și trimit noi etichete vecinilor lor, până când toate ruterele și-au însușit calea. De asemenea, pentru a garanta o calitate corespunzătoare a serviciilor, resursele pot fi alocate pe măsura construirii căii.

MPLS poate opera la mai multe niveluri deodată. La nivelul cel mai înalt, fiecare companie de telecomunicații poate fi privită ca un fel de metaruter, existând o cale prin metarutere de la sursă la destinație. Această cale poate folosi MPLS. Totuși, în cadrul rețelei fiecărei companii de telecomunicații poate fi folosit de asemenea MPLS, ducând la un al doilea nivel de etichetare. De fapt, un pachet poate transporta o întreagă stivă de etichete. Bitul *S* din fig. 5-41 permite unui ruter care șterge o etichetă să afle dacă au mai rămas alte etichete. Acesta are valoarea 1 pentru eticheta de la bază și 0 pentru toate celelalte etichete. În practică, această facilitate este folosită în principal la implementarea rețelelor virtuale private și a tunelelor recursive.

Deși ideile care au stat la baza MPLS-ului sunt simple, detaliile sunt extrem de complicate, cu multe variații și optimizări, așa că nu vom mai dezvolta acest subiect. Pentru mai multe informații, vezi (Davie și Rekhter, 2000; Lin et al., 2002; Pepelnjak și Guichard, 2001; și Wang, 2001).

5.5 INTERCONECTAREA REȚELELOR

Până în acest moment, am presupus implicit că există o singură rețea omogenă, în care fiecare mașină folosește același protocol la fiecare nivel. Din păcate, această presupunere este prea optimistă. Există mai multe tipuri de rețele, incluzând LAN-uri, MAN-uri și WAN-uri. La fiecare nivel se

folosesc pe larg numeroase protocoale. În secțiunile următoare vom analiza în detaliu problemele care apar când două sau mai multe rețele sunt conectate pentru a forma o **inter-rețea** (internet).

Există controverse considerabile pe tema întrebării dacă abundența actuală de tipuri de rețele este o condiție temporară, care va dispărea de îndată ce toată lumea își va da seama ce minunată este [completați cu tipul preferat de rețea], sau dacă este o caracteristică inevitabilă, dar permanentă, a lumii care va persista. Existența rețelelor de tipuri diferite înseamnă, invariabil, a avea protocoale diferite.

Credem că întotdeauna vor coexista o varietate de rețele diferite (și implicit de protocoale) din următoarele motive. În primul rând, numărul de rețele diferite instalate este mare. Aproape toate calculatoarele personale folosesc TCP/IP. Multe întreprinderi mari încă se bazează pe sisteme de calcul care folosesc SNA de la IBM. O parte importantă din companiile telefonice operează rețele ATM. Unele LAN-uri de calculatoare personale folosesc încă Novell NCP/IPX sau AppleTalk. Și, în final, în domeniul în plină dezvoltare al comunicațiilor fără fir există o mare varietate de protocoale. Această tendință va continua mult timp datorită problemelor de continuitate, noilor tehnologii și faptului că nu toți producătorii percep a fi în interesul lor posibilitatea clienților de a migra cu ușurință spre sistemul altui producător.

În al doilea rând, pe măsură ce calculatoarele și rețelele devin mai ieftine, nivelul la care se iau deciziile se mută în jos în organizație. Multe companii au o politică care stabilește că achizițiile de peste un milion de dolari trebuie să fie aprobate de conducerea superioară, achizițiile de peste 100.000 de dolari trebuie să fie aprobate de conducerea medie, dar achizițiile de până la 100.000 de dolari pot fi făcute de șefii de departamente fără nici o aprobare de mai sus. Aceasta poate duce ușor la instalarea unor stații de lucru UNIX care rulează TCP/IP în departamentul inginerie și a unor calculatoare Macintosh care folosesc AppleTalk în departamentul de marketing.

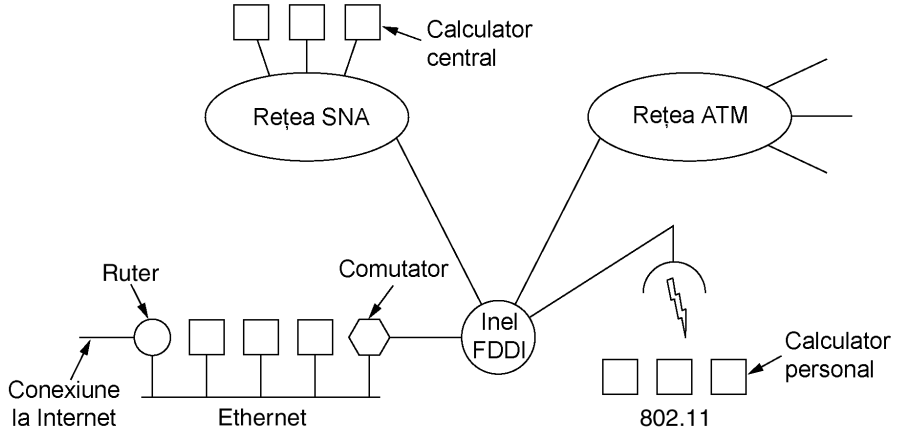


Fig. 5-42. Colecție de rețele interconectate.

În al treilea rând, rețele diferite (de exemplu ATM și fără fir) sunt bazate pe tehnologii radical diferite și nu ar trebui să surprindă că pe măsură ce apar dezvoltări hardware, se vor crea noi programe care să se potrivească cu noul hardware. De exemplu, casa medie de astăzi este asemănătoare cu biroul mediu de acum 10 ani; este plină de calculatoare care nu comunică unul cu celălalt. În viitor, ar putea fi un lucru obișnuit ca telefonul, televizorul și alte aparate electrocasnice să fie legate în rețea, pentru a permite controlul de la distanță. Această nouă tehnologie va aduce, fără dubii, noi rețele și noi protocoale.

Ca un exemplu al modului în care ar putea fi conectate rețele diferite, să considerăm exemplul din fig. 5-42. Aici vedem rețeaua unei companii cu mai multe sedii interconectate printr-o rețea ATM de mare întindere. La unul dintre sedii este folosită o coloană vertebrală (eng.: backbone) optică FDDI, prin care sunt conectate: o rețea Ethernet, un LAN fără fir 802.11 și rețeaua SNA de calculatoare centrale a centrului de date al corporației.

Scopul interconectării acestor rețele este de a permite utilizatorilor din orice rețea să comunice cu utilizatorii celorlalte rețele și de asemenea de a permite unui utilizator din orice rețea să acceseze date pe orice rețea. Realizarea acestui scop înseamnă trimiterea pachetelor dintr-o rețea în alta. Cum rețelele diferă deseori în puncte esențiale, transmiterea pachetelor dintr-o rețea în alta nu este întotdeauna ușoară, după cum vom vedea în continuare.

5.5.1 Prin ce diferă rețelele

Rețelele pot diferi în multe moduri. Unele dintre deosebiri, cum ar fi diferite tehnici de modulare sau formate de cadre, se găsesc la nivelul fizic și legătură de date. Aceste diferențe nu ne preocupă la acest nivel. În schimb, în fig. 5-43, enumerăm câteva din diferențele care pot apărea la nivelul rețea. Trecerea peste aceste diferențe face interconectarea rețelelor mult mai dificilă decât operarea într-o singură rețea.

Element	Câteva posibilități
Serviciu oferit	Orientat pe conexiuni față de cel fără conexiune
Protocol	IP, IPX, SNA, ATM, MPLS, AppleTalk etc.
Adresare	Plată (802) opusă celei ierarhice (IP)
Trimitere multiplă	Prezentă sau absentă (de asemenea, difuzarea totală)
Dimensiune pachet	Fiecare rețea își are propriul maxim
Calitatea serviciului	Poate fi prezentă sau absentă; multe tipuri diferite
Tratarea erorilor	Livrare fiabilă, ordonată sau neordonată
Controlul fluxului	Fereastră glisantă, controlul ratei, altele sau nimic
Controlul congestiei	Algoritmul găleții găurite, algoritmul găleții cu jeton, RED, pachete de șoc etc.
Securitate	Reguli de secretizare, criptare etc.
Parametri	Diferite limitări de timp, specificări ale fluxului etc.
Contabilizare	După timpul de conectare, după pachet, după octeți sau fără.

Fig. 5-43. Câteva din multele moduri în care pot diferi rețelele.

Când pachetele trimise de o sursă dintr-o rețea trebuie să tranziteze una sau mai multe rețele străine înainte de a ajunge în rețeaua destinație (care, de asemenea, poate fi diferită față de rețeaua sursă), pot apărea multe probleme la interfețele dintre rețele. Pentru început, atunci când pachetele dintr-o rețea orientată pe conexiuni trebuie să tranziteze o rețea fără conexiuni, poate interveni o reordonare a acestora, lucru la care emițătorul nu se așteaptă și căruia receptorul nu este pregătit să-i facă față. Vor fi necesare frecvente conversii de protocol, care pot fi dificile dacă funcționalitatea cerută nu poate fi exprimată. De asemenea, vor fi necesare conversii de adresă, ceea ce poate cere un sistem de catalogare. Trecerea pachetelor cu trimitere multiplă printr-o rețea care nu oferă trimitere multiplă necesită generarea de pachete separate pentru fiecare destinație.

Diferența dintre dimensiunile maxime ale pachetelor folosite de diferite rețele poate produce mari neplăceri. Cum veți trece un pachet de 8000 de octeți printr-o rețea a cărei dimensiune maximă este de 1500 de octeți? Diferențele în calitatea serviciilor sunt o problemă în momentul în care un

pachet care are constrângeri de livrare în timp real traversează o rețea care nu oferă nici o garanție de timp real.

Controlul erorilor, al fluxului și al congestiei diferă frecvent între rețele diferite. Dacă sursa și destinația așteaptă amândouă ca toate pachetele să fie livrate în ordine fără erori, dar o rețea intermediară elimină pachete ori de câte ori întrevede congestie la orizont, multe aplicații se vor comporta neprevăzut. Diferențele în ceea ce privește mecanismele de securitate, stabilirea parametrilor, regulile de contabilizare și chiar legile naționale referitoare la secrete pot, de asemenea, cauza probleme.

5.5.2 Cum pot fi conectate rețelele

Rețelele pot fi interconectate prin diferite dispozitive, așa cum s-a arătat în cap.4. Să revedem pe scurt acest material. La nivelul fizic, rețelele pot fi conectate prin repetoare sau noduri (eng.: hubs), care doar transferă biții între două rețele identice. Acestea sunt în marea lor majoritate dispozitive analogice și nu cunosc protocoalele numerice (doar regenerează semnale).

Cu un nivel mai sus întâlnim punțile și comutatoarele, care operează la nivelul legăturii de date. Acestea acceptă cadre, examinează adresele MAC și retransmit cadrele către o rețea diferită, efectuând traduceri de protocol minore, ca de exemplu de la Ethernet la FDDI sau la 802.11.

La nivelul rețea avem rutere care pot conecta două rețele. Dacă două rețele au niveluri rețea diferite, ruterul poate fi capabil să transforme formatul pachetelor, cu toate că astfel de situații sunt din ce în ce mai rare. Un ruter care poate trata mai multe protocoale este numit **ruter multiprotocol** (eng. multiprotocol router).

La nivelul transport întâlnim porți de transport, care por realiza interfața între două conexiuni de transport. De exemplu, o poartă de transport poate permite pachetelor să treacă între o rețea TCP și o rețea SNA, care are un protocol de transport diferit, făcând legătura între o conexiune TCP și o conexiune SNA.

În sfârșit, la nivelul aplicație, porțile de aplicație traduc semnificația mesajelor. De exemplu, punțile dintre poșta electronică din Internet (RFC 822) și poșta electronică X 400 trebuie să analizeze mesajele și să schimbe diferite câmpuri din antete.

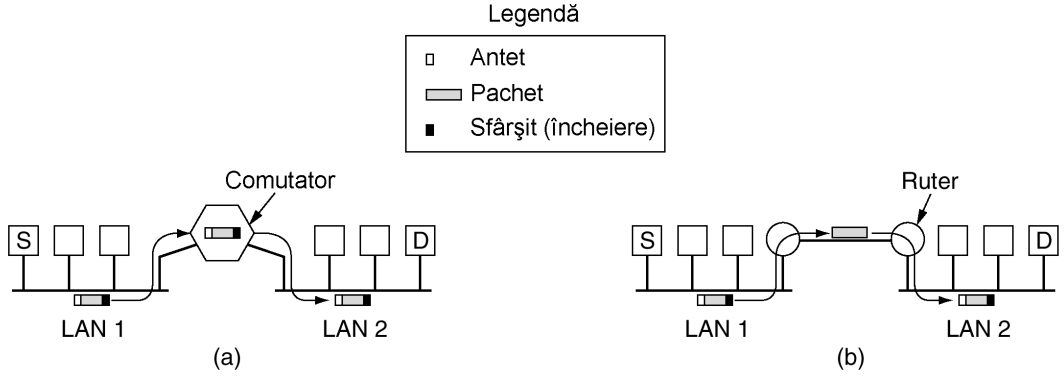


Fig. 5-44. (a) Două rețele Ethernet conectate printr-un comutator. (b) Două rețele Ethernet conectate prin rutere.

În acest capitol ne vom concentra pe interconectarea rețelelor la nivelul rețea. Pentru a vedea prin ce diferă de comutarea la nivelul legăturii de date, să examinăm fig. 5-44. În fig. 5-44(a) mașina

sursă, *S*, dorește să trimită un pachet mașinii destinație, *D*. Aceste mașini se află în rețele Ethernet diferite, conectate printr-un comutator. *S* încapsulează pachetul într-un cadru și îl trimite spre destinație. Cadrul ajunge la comutator, care determină, analizând adresa MAC, destinația cadrului, reprezentată de LAN 2. Comutatorul nu face decât să preia cadrul din LAN 1 și să îl pună pe LAN 2.

Să considerăm acum aceeași situație, dar cu două rețele Ethernet conectate printr-o pereche de rutere, în loc de comutator. Ruterele sunt conectate printr-o linie punct-la-punct, posibil o linie închiriată de mai mulți kilometri lungime. Acum cadrul este preluat de ruter, iar pachetul este extras din câmpul de date al cadrului. Ruterul examinează adresa din pachet (de exemplu o adresă IP) și o caută în tabela sa de rutare. Pe baza acestei adrese decide să trimită pachetul la ruterul aflat la distanță, eventual încapsulat într-un alt tip de cadru, în funcție de protocolul liniei. La celălalt capăt pachetul este pus în câmpul de date al unui cadru Ethernet și este depus pe LAN 2.

O diferență esențială între cazul cu comutator (sau punte) și cazul cu rutere este următoarea. În cazul cu comutator (sau punte) este transportat întregul cadru, pe baza adresei MAC. În cazul unui ruter pachetul este extras din cadru, iar adresa din pachet este utilizată pentru a decide unde să fie trimis. Comutatoarele nu trebuie să înțeleagă protocolul nivelului rețea, dar ruterele da.

5.5.3 Circuite virtuale concatenate

Sunt posibile două stiluri de interconectare a rețelelor: o concatenare orientată pe conexiuni a subrețelelor cu circuite virtuale și un stil datagrame inter-rețea. Vom examina pe rând aceste variante, dar înainte de aceasta, un avertisment. În trecut, marea majoritate a rețelelor (publice) erau orientate pe conexiune (și releul de cadre, SNA, 802.16 și ATM încă sunt). Apoi, odată cu acceptarea rapidă a Internetului, datagramele au venit la modă. Totuși ar fi o greșeală să credem că datagramele vor fi folosite la nesfârșit. În acest domeniu singurul lucru etern este schimbarea. Odată cu creșterea importanței rețelelor multimedia, este probabil ca orientarea pe conexiune să revină într-o formă sau alta, deoarece este mai ușor să se garanteze calitatea serviciului cu conexiuni, decât fără ele. De aceea în continuare vom dedica spațiu orientării pe conexiune.

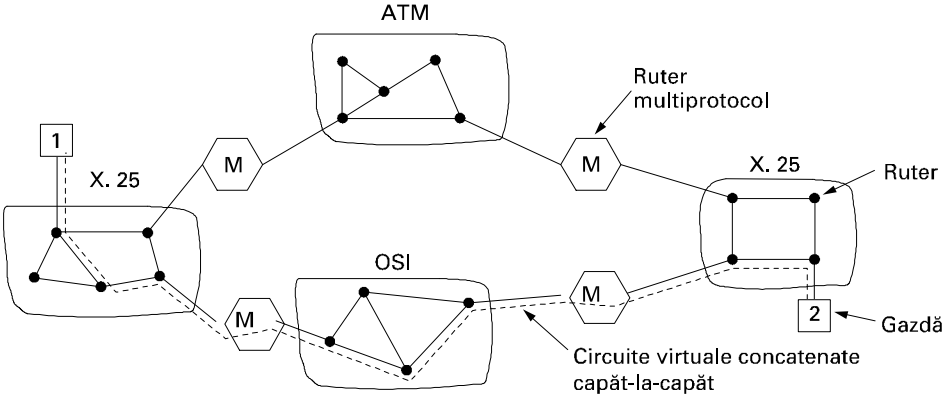


Fig. 5-45. Interconectarea rețelelor folosind circuite virtuale concatenate.

În modelul circuitelor virtuale concatenate, arătat în fig. 5-45, o conexiune către o gazdă dintr-o rețea îndepărtată este stabilită într-un mod similar cu modul în care sunt stabilite conexiunile în cazul normal. Subrețeaua observă că destinația este îndepărtată și construiește un circuit virtual spre

ruterul aflat cel mai aproape de rețeaua destinație. Apoi construiește un circuit virtual de la acel ruter la o **poartă** externă (un ruter multiprotocol). Această poartă înregistrează existența circuitului virtual în tabelele sale și continuă să construiască un alt circuit virtual către un ruter din următoarea subrețea. Acest proces continuă până când se ajunge la gazda destinație.

O dată ce pachetele de date încep să circule de-a lungul căii, fiecare poartă retransmite pachetele primite, făcând, după nevoie, conversia între formatele pachetelor și numerele de circuite virtuale. Evident, toate pachetele de date trebuie să traverseze aceeași secvență de porți, deci ajung în ordine.

Caracteristica esențială a acestei abordări este că se stabilește o secvență de circuite virtuale de la sursă, prin una sau mai multe porți, până la destinație. Fiecare poartă menține tabele spunând ce circuite virtuale o traversează, unde vor fi dirijate și care este numărul noului circuit virtual.

Această schemă funcționează cel mai bine atunci când toate rețelele au, în mare, aceleași proprietăți. De exemplu, dacă toate garantează livrarea sigură a pachetelor de la nivelul rețea, atunci, exceptând un accident de-a lungul căii, fluxul de la sursă la destinație va fi de asemenea sigur. Similar, dacă nici una din ele nu garantează livrarea sigură, atunci concatenarea circuitelor virtuale nu este nici ea sigură. Pe de altă parte, dacă mașina sursă este într-o rețea care garantează livrarea sigură, dar una din rețelele intermediare poate pierde pachete, concatenarea schimbă fundamental natura serviciului.

Circuitele virtuale concatenate sunt, de asemenea, uzuale la nivelul transport. În particular, este posibil să se construiască o conductă de biți folosind, să spunem, SNA, care se termină într-o poartă și având o conexiune TCP de la această poartă la poarta următoare. În acest mod, un circuit virtual capăt-la-capăt poate fi construit acoperind diferite rețele și protocoale.

5.5.4 Interconectarea rețelelor fără conexiuni

Modelul alternativ de interconectare este modelul datagramă, prezentat în fig. 5-46. În acest model, singurul serviciu pe care nivelul rețea îl oferă nivelului transport este capacitatea de a injecta datagrame în subrețea în speranța că totul va merge bine. Nu există nici o noțiune de circuit virtual la nivelul rețea și uitați de concatenarea lor. Acest model nu necesită ca toate pachetele care aparțin unei conexiuni să traverseze aceeași secvență de porți. În fig. 5–46 sunt ilustrate datagramele de la gazda 1 pentru gazda 2, care urmează rute diferite prin rețeaua de interconectare. O decizie de dirijare este luată separat pentru fiecare pachet, eventual în funcție de traficul din momentul în care este trimis pachetul. Această strategie poate utiliza rute multiple și atinge astfel o capacitate mai mare decât modelul circuitelor virtuale concatenate. Pe de altă parte, nu există nici o garanție că pachetele ajung la destinație în ordine, presupunând că vor ajunge.

Modelul din fig. 5-46 nu este așa de simplu precum pare. Pe de o parte, dacă fiecare rețea are propriul protocol de nivel rețea, nu este posibil ca un pachet dintr-o rețea să tranziteze alta. Se pot imagina rutere multiprotocol care încearcă să convertească dintr-un format în altul, dar, în afara cazului în care cele două formate sunt strâns înrudite având aceleași câmpuri de informație, aceste conversii vor fi incomplete și deseori sortite eșecului. Din acest motiv, arareori se încearcă conversii.

O a doua problemă și mai serioasă este adresarea. Să ne imaginăm un caz simplu: o gazdă din Internet încearcă să trimită un pachet IP către o gazdă dintr-o rețea adiacentă SNA. Adresele IP și SNA sunt diferite. Ar trebui stabilită o corespondență între adresele IP și SNA și invers. În plus, ceea ce se poate adresa este diferit. În IP, gazdele (de fapt plăcile de interfață) au adrese. În SNA, pot avea adrese și alte entități în afară de gazde (de exemplu echipamente hardware). În cel mai bun caz, cineva ar trebui să mențină o bază de date a tuturor corespondențelor, dar ar fi o sursă constantă de necazuri.

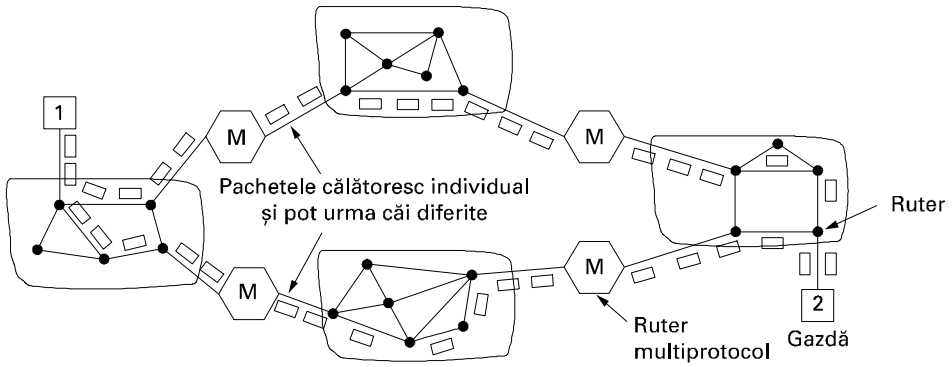


Fig. 5-46. O interconectare fără conexiuni.

O altă idee este de a proiecta un pachet universal „inter-rețea" și a obliga toate ruterele să-l recunoască. Această abordare este, de fapt, chiar IP-ul - un pachet proiectat ca să fie purtat prin mai multe rețele. Bineînțeles se poate întâmpla ca IPv4 (protocolul actual din Internet) să scoată de pe piață toate celelalte protocoale, IPv6 (viitorul protocol Internet) să nu prindă și să nu mai fie inventat nimic nou, dar istoria sugerează altceva. Este dificil ca toată lumea să fie de acord cu un singur format, atunci când companiile consideră că este în interesul lor să aibă formate proprii pe care să le controleze.

Să recapitulăm pe scurt cele două moduri prin care se poate aborda interconectarea rețelelor. Modelul circuitelor virtuale concatenate are, în esență, aceleași avantaje ca folosirea circuitelor virtuale într-o singură subrețea: zonele tampon pot fi rezervate în prealabil, poate fi garantată secvențialitatea, pot fi folosite antete scurte, iar necazurile cauzate de pachetele duplicate întârziate pot fi evitate.

El are, de asemenea, și dezavantaje: spațiul în tabele necesar în fiecare ruter pentru fiecare conexiune deschisă, lipsa unei dirijări alternative pentru a evita zonele congestionate și vulnerabilitatea la defectarea ruterelor de pe parcurs. De asemenea, are dezavantajul de a fi dificil, dacă nu imposibil, de implementat în cazul în care una din rețelele implicate este rețea nesigură de tip datagramă.

Proprietățile abordării interconectării rețelelor prin datagrame sunt în mare parte aceleași ca și cele ale subrețelelor de tip datagramă: potențial de congestionare mai mare, dar de asemenea potențial mai mare de adaptare la congestionări, robustețe în cazul defectării ruterelor și lungime mai mare necesară pentru antete. Într-o inter-rețea sunt posibili diferiți algoritmi de dirijare adaptivă, așa cum sunt în cadrul unei singure rețele de tip datagramă. Un avantaj major al abordării interconectării rețelelor prin datagrame este că aceasta poate fi folosită peste subrețele care nu folosesc circuite virtuale în interior. Multe LAN-uri, rețele mobile (de exemplu flotele aeriene și navale) și chiar unele WAN-uri intră în această categorie. Când o inter-rețea include una dintre acestea, apar probleme serioase dacă strategia de interconectare a rețelelor este bazată pe circuite virtuale.

5.5.5 Trecerea prin tunel

Rezolvarea cazului general de interconectare a două rețele diferite este extrem de dificilă. Cu toate acestea, există un caz special uzual care este gestionabil. Acest caz apare când gazdele sursă și destinație sunt în același tip de rețea, dar între ele există o rețea diferită. Ca exemplu, gândiți-vă la o bancă internațională cu o rețea TCP/IP bazată pe Ethernet la Paris, o rețea TCP/IP bazată pe Ethernet la Londra și o rețea non-IP de mare întindere (de exemplu ATM), așa cum este prezentat în fig. 5-47.

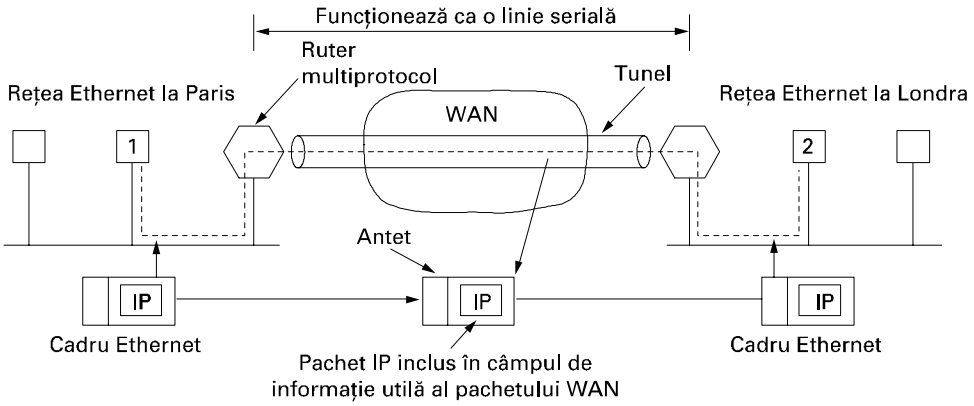


Fig. 5-47. Utilizarea tunelului pentru un pachet trimis de la Paris la Londra.

Soluția acestei probleme este o tehnică numită **trecerea prin tunele**. Pentru a trimite un pachet IP la gazda 2, gazda 1 construiește pachetul conținând adresa IP a gazdei 2, îl inserează într-un cadru Ethernet adresat ruterului multiprotocol parizian și apoi îl trimite în rețeaua Ethernet. Când ruterul multiprotocol primește cadrul, extrage pachetul IP, îl inserează în câmpul informație utilă al pachetului de nivel rețea WAN, pachet pe care îl adresează cu adresa ruterului multiprotocol londonez. Când ajunge acolo, ruterul londonez extrage pachetul IP și îl trimite gazdei 2 în interiorul unui cadru Ethernet.

WAN-ul poate fi văzut ca un mare tunel ce se întinde de la un ruter multiprotocol la altul. Pachetul IP doar traversează tunelul de la un capăt la altul, așezat confortabil în frumosul său lăcaș. El nu trebuie să aibă deloc grijă de comportarea la nivel WAN. Și nici gazdele din oricare Ethernet. Numai ruterul multiprotocol trebuie să înțeleagă și pachete IP și WAN. Ca rezultat, întreaga distanță de la mijlocul unui ruter multiprotocol până la mijlocul celuilalt acționează ca o linie serială.

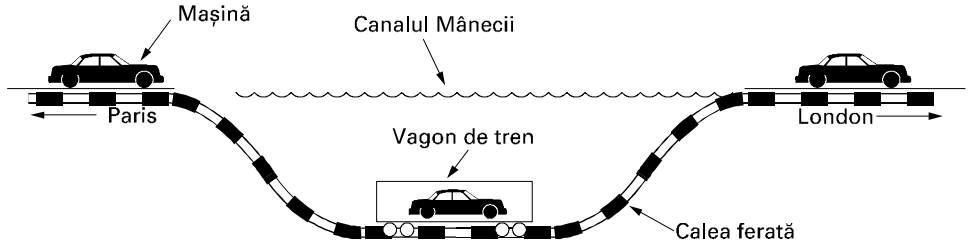


Fig. 5-48. Trecerea unui automobil din Franța în Anglia, prin tunel.

O analogie poate face utilizarea tunelelor mai clară. Considerați o persoană conducând mașina de la Paris la Londra. În Franța, mașina se deplasează sub acțiunea propriei puteri, dar când ajunge la Canalul Mânecii, este încărcată într-un tren de mare viteză și transportată în Anglia prin Chunnel[6] (mașinile nu pot circula prin Chunnel). Efectiv, mașina este purtată ca informație utilă, așa cum este prezentat în fig. 5-48. La capătul englez, mașina este eliberată pe drumurile engleze și continuă să se

[6] N.T. Chunnel, ce provine de la Channel (canal) și Tunnel (tunel) este denumirea dată în engleză tunelului de sub Canalul Mânecii.

deplaseze, din nou cu propria putere. Utilizarea tunelelor printr-o rețea necunoscută funcționează în același mod.

5.5.6 Dirijarea în rețele interconectate

Dirijarea printr-o rețea interconectată este similară cu dirijarea într-o singură subrețea, dar cu câteva complicații în plus. Să considerăm, de exemplu, interconectarea rețelelor din fig. 5-49(a) în care cinci rețele sunt conectate prin șase rutere (posibil multiprotocol). Crearea unui graf ca model al acestei situații este complicată de faptul că fiecare ruter poate accesa direct (mai clar, poate trimite pachete la) orice alt ruter conectat la orice rețea cu care este conectat. De exemplu, B din fig. 5-49(a) poate accesa direct A și C prin rețeaua 2 și de asemenea D prin rețeaua 3. Aceasta duce la graful din fig. 5-49(b).

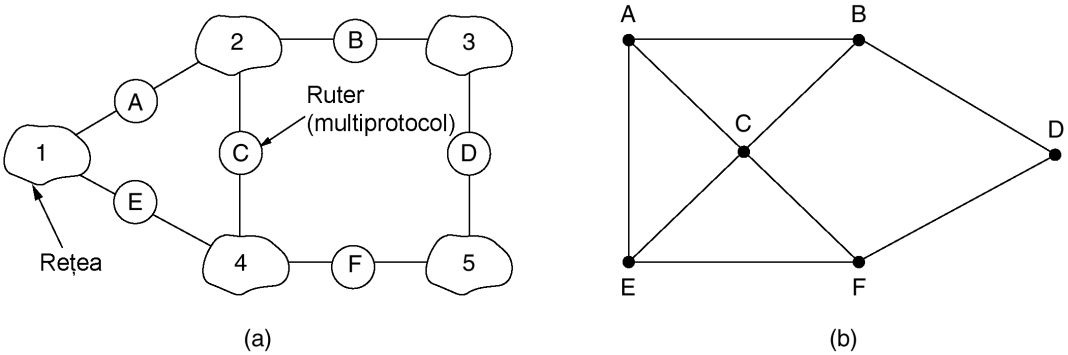


Fig. 5-49. (a) O interconectare de rețele. (b) Un graf al interconectării rețelelor.

O dată ce graful a fost construit, pe mulțimea de rutere multiprotocol pot fi aplicați algoritmii de dirijare cunoscuți, cum ar fi algoritmii de tip vectori distanță sau bazați pe starea legăturii. Aceasta duce la un algoritm de dirijare în doi pași: în interiorul fiecărei rețele se folosește un **protocol de poartă interioară (internal gateway protocol)**, dar între rețele se folosește un **protocol de poartă exterioară (exterior gateway protocol)** („poartă" este un termen mai vechi pentru „ruter„). De fapt, din moment ce fiecare rețea este independentă, ele pot folosi algoritmi diferiți. Deoarece fiecare rețea dintr-o interconectare de rețele este independentă de toate celelalte, este deseori referită ca un **sistem autonom (autonomous system-AS)**.

Un pachet tipic inter-rețele pornește din LAN-ul său adresat ruterului multiprotocol local (în antetul nivelului MAC). După ce ajunge acolo, codul de la nivelul rețea decide cărui ruter multiprotocol să-i trimită pachetul, folosind propriile tabele de dirijare. Dacă la acel ruter se poate ajunge folosind protocolul de rețea nativ al pachetului, atunci este trimis direct acelui ruter. Altfel, este trimis utilizând tunele, încapsulat în protocolul cerut de rețeaua intermediară. Acest proces este repetat până când pachetul ajunge în rețeaua destinație.

Una din diferențele dintre dirijarea inter-rețele și dirijarea intra-rețele este că dirijarea inter-rețele poate necesita deseori traversarea granițelor internaționale. Intră în joc legi diferite, cum ar fi legile suedeze despre secretul strict care se referă la exportarea din Suedia de date personale referitoare la cetățenii suedezi. Un alt exemplu este legea canadiană care spune că traficul de date care pornește din Canada și se termină în Canada nu poate părăsi țara. Această lege înseamnă că traficul

din Windsor, Ontario spre Vancouver nu poate fi dirijat prin apropiatul Detroit, chiar dacă această rută este cea mai rapidă și cea mai ieftină.

O altă diferență între dirijarea internă și cea externă este costul. Într-o singură rețea, se aplică în mod normal un singur algoritm de taxare. Cu toate acestea, diferite rețele pot avea administrații diferite și o cale poate fi mai ieftină decât alta. Similar, calitatea serviciului oferit de rețele diferite poate fi diferită și acesta poate fi un motiv pentru alegerea unei căi în defavoarea alteia.

5.5.7 Fragmentarea

Fiecare rețea impune o anumită dimensiune maximă pentru pachetele sale. Aceste limite au diferite cauze, printre care:

1. Hardware (de exemplu, dimensiunea unui cadru Ethernet).
2. Sistemul de operare (de exemplu, toate zonele tampon au 512 octeți).
3. Protocoale (de exemplu, numărul de biți din câmpul lungime al pachetului).
4. Concordanța cu unele standarde (inter)naționale.
5. Dorința de a reduce la un anumit nivel retransmisiile provocate de erori.
6. Dorința de a preveni ocuparea îndelungată a canalului de către un singur pachet.

Rezultatul tuturor acestor factori este că proiectanții de rețele nu au libertatea de a alege dimensiunea maximă a pachetelor oricum ar dori. Informația utilă maximă variază de la 8 octeți (celulele ATM) la 65515 octeți (pachetele IP), cu toate că dimensiunea pachetelor la nivelurile mai înalte este deseori mai mare.

O problemă evidentă apare când un pachet mare vrea să traverseze o rețea în care dimensiunea maximă a pachetului este prea mică. O soluție este să ne asigurăm din capul locului că problema nu apare. Cu alte cuvinte, inter-rețeaua trebuie să utilizeze un algoritm de dirijare care evită transmiterea pachetelor prin rețele în care pachetele nu pot fi manevrate. Cu toate acestea, această soluție nu este de fapt nici o soluție. Ce se întâmplă dacă pachetul sursă original este prea mare pentru a fi manevrat de rețeaua destinație? Algoritmul de dirijare nu are cum să evite destinația.

În esență, singura soluție a problemei este de a permite porților să spargă pachetele în **fragmente**, trimițând fiecare pachet ca un pachet inter-rețea separat. Cu toate acestea, așa cum știe orice părinte al unui copil mic, convertirea unui obiect mare în fragmente mici este considerabil mai ușoară decât procesul invers. (Fizicienii au dat chiar un nume acestui efect: legea a doua a termodinamicii.) Rețelele cu comutare de pachete au, de asemenea, probleme în îmbinarea fragmentelor.

Există două strategii opuse pentru reconstituirea pachetului original din fragmente. Prima strategie este de a face fragmentarea cauzată de o rețea cu „pachete mici” transparentă pentru toate rețelele succesive prin care pachetul trebuie să treacă pe calea către destinația finală. Această opțiune este prezentată în fig. 5-50(a). În această strategie, rețeaua cu pachete mici are porți (cel mai probabil, rutere specializate) către celelalte rețele. Când un pachet supradimensionat ajunge la poartă, poarta îl sparge în fragmente. Fiecare fragment este adresat aceleiași porți de ieșire, unde piesele sunt recombinate. În acest mod, trecerea printr-o rețea cu pachete mici a devenit transparentă. Rețelele următoare nici măcar nu sunt conștiente de fragmentarea făcută. Rețelele ATM, de exemplu, au hardware special pentru a oferi fragmentarea transparentă a pachetelor în celule și apoi reasamblarea celulelor în pachete. În lumea ATM, fragmentarea este numită segmentare; conceptul este același, dar diferă unele detalii.

Fragmentarea transparentă este simplă, dar are câteva probleme. Un motiv este că poarta de ieșire trebuie să știe când a primit toate piesele, așa încât în fiecare pachet trebuie inclus fie un câmp contor, fie un bit „sfârșit-de-pachet„. Un alt motiv este că toate pachetele trebuie să iasă prin aceeași poartă. Performanțele se pot degrada nepermițând ca unele pachete să urmărească o cale către destinația finală și alte pachete o cale diferită. O ultimă problemă este timpul suplimentar necesar pentru reasamblarea și apoi refragmentarea repetată a unui pachet mare care traversează o serie de rețele cu pachete mici. ATM necesită fragmentare transparentă.

Cealaltă strategie de fragmentare este de a nu recombina fragmentele la nici o poartă intermediară. O dată ce un pachet a fost fragmentat, fiecare fragment este tratat ca și cum ar fi un pachet original. Toate fragmentele sunt trecute printr-o poartă (sau porți) de ieșire, așa cum se arată în fig. 5-50(b). Recombinarea are loc doar la gazda destinație. Așa funcționează IP-ul.

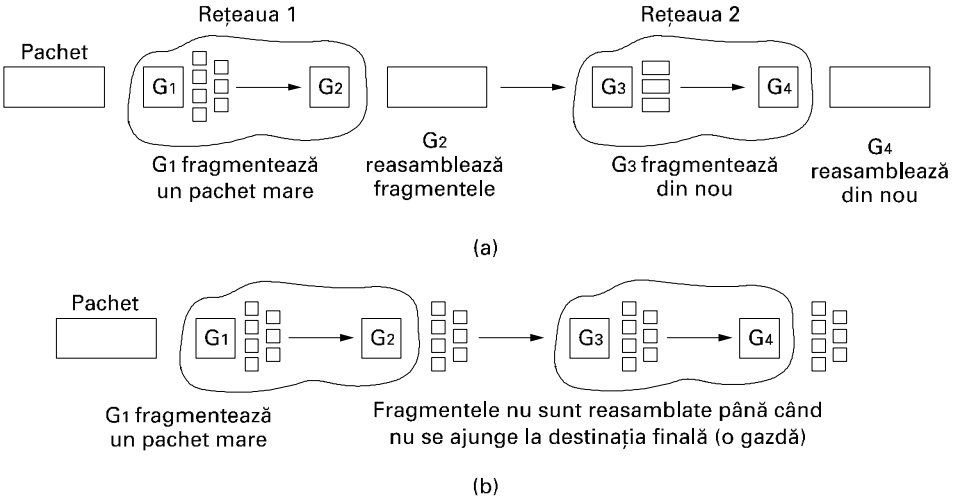


Fig. 5-50. (a) Fragmentare transparentă. (b) Fragmentare netransparentă.

Fragmentarea netransparentă are, de asemenea, unele probleme. De exemplu, necesită ca fiecare gazdă să fie capabilă să facă reasamblarea. Încă o problemă este că atunci când se fragmentează un pachet mare, supraîncărcarea crește, deoarece fiecare fragment trebuie să aibă un antet. Pe câtă vreme în prima metodă această supraîncărcare dispare de îndată ce se iese din rețeaua cu pachete mici, în această metodă supraîncărcarea rămâne pentru restul călătoriei. Cu toate acestea, un avantaj al acestei metode este că se pot folosi mai multe porți de ieșire și se pot obține performanțe mai bune. Desigur, dacă se folosește modelul circuitelor virtuale concatenate, acest avantaj nu este de nici un folos.

Când un pachet este fragmentat, fragmentele trebuie numerotate astfel încât șirul inițial de date să poată fi reconstituit. O metodă de numerotare a fragmentelor este bazată pe un arbore. Dacă pachetul 0 trebuie descompus, bucățile sunt numite 0.0, 0.1, 0.2 etc. Dacă aceste fragmente trebuie la rândul lor, să fie fragmentate mai târziu, bucățile sunt numerotate 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 etc. Dacă în antet au fost rezervate suficiente câmpuri pentru cel mai rău caz și nu sunt generate duplicate în altă parte, această schemă este suficientă pentru a asigura că toate bucățile pot fi corect reasamblate la destinație, neavând importanță ordinea în care ajung.

Cu toate acestea, dacă o singură rețea pierde sau elimină pachete, apare necesitatea unor retransmisii capăt-la-capăt, cu efecte nefericite pentru schema de numerotare. Să presupunem că un pa-

chet de 1024 biți este inițial fragmentat în patru fragmente de dimensiune egală, 0.0, 0.1, 0.2 și 0.3. Fragmentul 0.1 este pierdut, dar toate celelalte părți ajung la destinație. În cele din urmă, la sursă apare o depășire de timp și aceasta retransmite pachetul original. Numai că de această dată intervine legea lui Murphy și calea trece printr-o rețea cu limita de 512 octeți, deci sunt generate două fragmente. Când noul fragment 0.1 ajunge la destinație, receptorul va considera că toate cele patru bucăți au ajuns și va reconstrui pachetul incorect.

Un sistem de numerotare diferit (și mai bun) este ca protocolul de interconectare a rețelelor să definească o dimensiune de fragment elementar suficient de mică ca fragmentul elementar să poată trece prin orice rețea. Când un pachet este fragmentat, toate bucățile sunt egale cu dimensiunea fragmentului elementar, cu excepția ultimului, care poate fi mai scurt. Un pachet inter-rețea poate conține mai multe fragmente, din motive de eficiență. Antetul inter-rețea trebuie să ofere numărul original al pachetului și numărul fragmentului (sau primului fragment) elementar conținut în pachet. Ca de obicei, trebuie să existe un bit care să indice că ultimul fragment elementar conținut în pachetul inter-rețea este ultimul din pachetul original.

Această abordare necesită două câmpuri de secvență în antetul inter-rețea: numărul pachetului original și numărul fragmentului. Există clar un compromis între dimensiunea fragmentului elementar și numărul de biți din numărul fragmentului. Deoarece dimensiunea fragmentului elementar este presupusă a fi acceptabilă pentru toate rețelele, fragmentările ulterioare ale unui pachet inter-rețea conținând câteva fragmente nu cauzează probleme. În acest caz, limita extremă este reprezentată de un fragment elementar de un bit sau octet, numărul de fragment fiind reprezentat de deplasamentul bitului sau octetului în cadrul pachetului original, așa cum se arată în fig. 5-51.

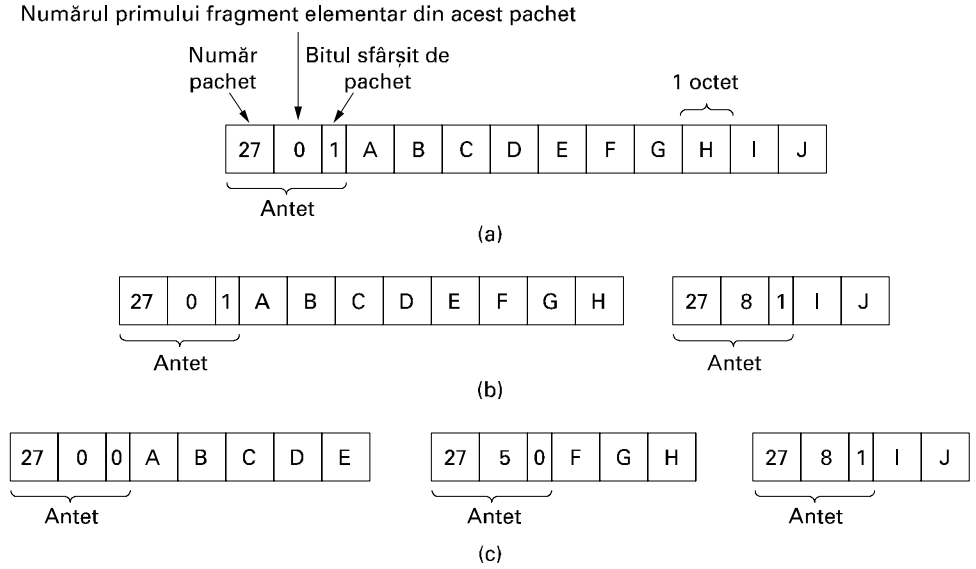


Fig. 5-51. Fragmentarea când dimensiunea datelor elementare este 1 octet. (a) Pachetul original, conținând 10 octeți de date. (b) Fragmente după trecerea printr-o rețea cu dimensiunea maximă a pachetului de 8 octeți. (c) Fragmente după trecerea printr-o poartă cu dimensiunea de 5.

Unele protocoale inter-rețea extind această metodă și consideră întreaga transmisie pe un circuit virtual ca fiind un pachet gigant, așa încât fiecare fragment conține numărul absolut de octet al primului octet din fragment.

5.6 NIVELUL REȚEA ÎN INTERNET

Înainte de a intra în detaliile nivelului rețea din Internet merită studiate principiile care au ghidat proiectarea lui în trecut și l-au făcut succesul care este astăzi. În ziua de azi, în prea multe cazuri oamenii par să le fi uitat. Aceste principii sunt enumerate și discutate în RFC 1958, care merită citit (și ar trebui să fie obligatoriu pentru toți proiectanții de protocoale – cu un examen final la sfârșit). Acest RFC se bazează în mare parte pe idei care se găsesc în (Clark, 19 și Saltzer, 1984). Vom rezuma ceea ce considerăm a fi cele 10 principii (de la cel mai important la cel mai puțin important).

1. **Fiți siguri că funcționează**. Nu finalizați proiectarea sau standardul până când mai multe prototipuri nu au comunicat cu succes unele cu altele. De prea multe ori proiectanții întâi scriu un standard de 1000 de pagini, primesc aprobarea pentru el și apoi descoperă că acest standard este eronat și nu funcționează. Apoi scriu versiunea 1.1 a standardului. Nu așa se face.
2. **Menține-l simplu**. Când există îndoieli, folosiți soluția mai simplă. William de Occam a enunțat acest principiu (briciul lui Occam) în secolul al 14-lea. Reformulat în termeni moderni: împotriviți-vă caracteristicilor suplimentare. Dacă o caracteristică nu este absolut necesară, nu o includeți, mai ales dacă același efect poate fi obținut prin combinarea altor caracteristici.
3. **Faceți alegeri clare**. Dacă există mai multe moduri de a realiza același lucru, alegeți unul. A avea două sau mai multe moduri de a rezolva un lucru înseamnă să se caute necazul cu lumânarea. Standardele conțin de obicei multe opțiuni sau moduri sau parametri pentru că anumite grupări importante susțin că modul lor este cel mai bun. Proiectanții ar trebui să reziste la această tendință. Spuneți nu.
4. **Exploatați modularitatea**. Acest principiu conduce direct la ideea de a avea stive de protocoale, fiecare nivel fiind independent de toate celelalte. În acest mod, dacă circumstanțele cer ca un modul sau nivel să fie schimbat, celelalte nu vor fi afectate.
5. **Așteptați-vă la medii eterogene**. În orice rețea mare vor apărea diferite tipuri de hardware, posibilități de transmisie și aplicații. Pentru a le trata pe toate, proiectarea rețelei trebuie să fie simplă, generală și flexibilă.
6. **Evitați opțiuni sau parametri statici**. Dacă un parametru nu poate fi evitat (de exemplu dimensiunea maximă a unui pachet), este mai bine ca transmițătorul și receptorul să negocieze o valoare decât să se definească valori fixe.
7. **Căutați o proiectare bună; nu este necesar să fie perfectă**. Deseori proiectanții au un proiect bun care nu poate trata un caz mai ciudat. Decât să strice tot proiectul, proiectanții ar trebui să-l păstreze și să transfere povara rezolvării acelui caz celor cu cerințe ciudate.

8. **Fiți stricți atunci când trimiteți și toleranți atunci când recepționați**. Cu alte cuvinte, trimiteți numai pachete care sunt în deplină conformitate cu standardul, dar așteptați-vă ca pachetele recepționate să nu fie în deplină conformitate cu standardul și încercați să le folosiți.
9. **Gândiți-vă la scalabilitate**. Dacă sistemul trebuie să suporte milioane de gazde și efectiv miliarde de utilizatori, nu se poate accepta nici un fel de bază de date centralizată, iar încărcarea trebuie distribuită cât mai uniform posibil folosind resursele disponibile.
10. **Luați în considerare performanțele și costurile**. Dacă o rețea are performanțe slabe sau prețuri exorbitante, nu o va folosi nimeni.

Să lăsăm acum principiile generale și să începem să studiem detaliile nivelului rețea din Internet. La nivelul rețea, Internet-ul poate fi văzut ca o colecție de subrețele sau **sisteme autonome** (eng.: **Autonomous Systems - AS**) care sunt interconectate. Nu există o structură reală, dar există câteva coloane vertebrale majore. Acestea sunt construite din linii de înaltă capacitate și rutere rapide. La coloanele vertebrale sunt atașate rețelele regionale (de nivel mediu), iar la aceste rețele regionale sunt atașate LAN-urile din multe universități, companii și furnizori de servicii Internet. O schiță a acestei organizări cvasi-ierarhice este dată în fig. 5-52.

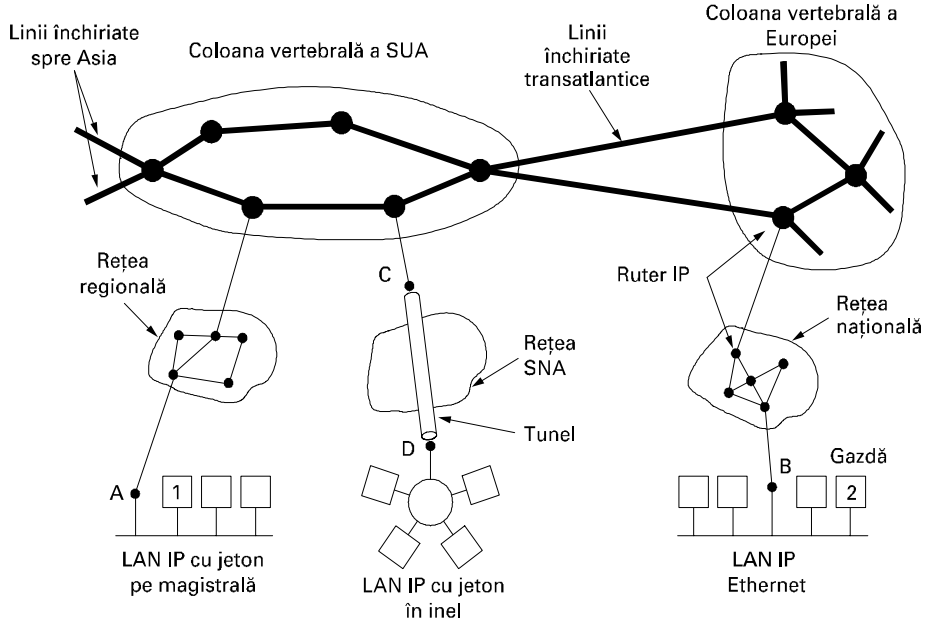


Fig. 5-52. Internet-ul este o colecție de multe rețele interconectate.

Liantul care ține Internet-ul la un loc este protocolul de nivel rețea, numit **IP** (**Internet Protocol** - protocolul Internet). Spre deosebire de protocoalele mai vechi de nivel rețea, acesta a fost proiectat de la început având în vedere interconectarea rețelelor. O metodă bună de a gândi nivelul rețea este aceasta. Sarcina lui este de a oferi cel mai bun mod posibil (adică negarantat) de a transporta datagrame de la sursă la destinație, fără a ține seama dacă aceste mașini sunt sau nu în aceeași rețea sau dacă între ele există sau nu alte rețele.

Comunicația în Internet funcționează după cum urmează. Nivelul transport preia șiruri de date și le sparge în datagrame. În teorie, datagramele pot avea fiecare până la 64 KB, dar în practică, ele nu depășesc 1500 octeți (pentru a intra într-un cadru Ehternet) . Fiecare datagramă este transmisă prin Internet, fiind eventual fragmentată în unități mai mici pe drum. Când toate bucățile ajung în sfârșit la mașina destinație, ele sunt reasamblate de nivelul rețea în datagrama originală. Datagrama este apoi pasată nivelului transport, care o inserează în șirul de intrare al procesului receptor. După cum se poate vedea în fig. 5-52 un pachet transmis de gazda 1 trebuie să traverseze șase rețele pentru a ajunge la gazda 2. În practică se trece de obicei prin mult mai mult decât șase rețele.

5.6.1 Protocolul IP

Un loc potrivit pentru a porni studiul nostru despre nivelul rețea în Internet este însuși formatul datagramelor IP. O datagramă IP constă dintr-o parte de antet și o parte de text. Antetul are o parte fixă de 20 de octeți și o parte opțională cu lungime variabilă. Formatul antetului este prezentat în fig. 5-53. El este transmis în ordinea *big endian* (cel mai semnificativ primul): de la stânga la dreapta, începând cu bitul cel mai semnificativ al câmpului *Versiune*. (Procesorul SPARC este de tip *big endian*; Pentium este de tip *little endian* - cel mai puțin semnificativ primul). Pe mașinile de tip *little endian*, este necesară o conversie prin program atât la transmisie cât și la recepție.

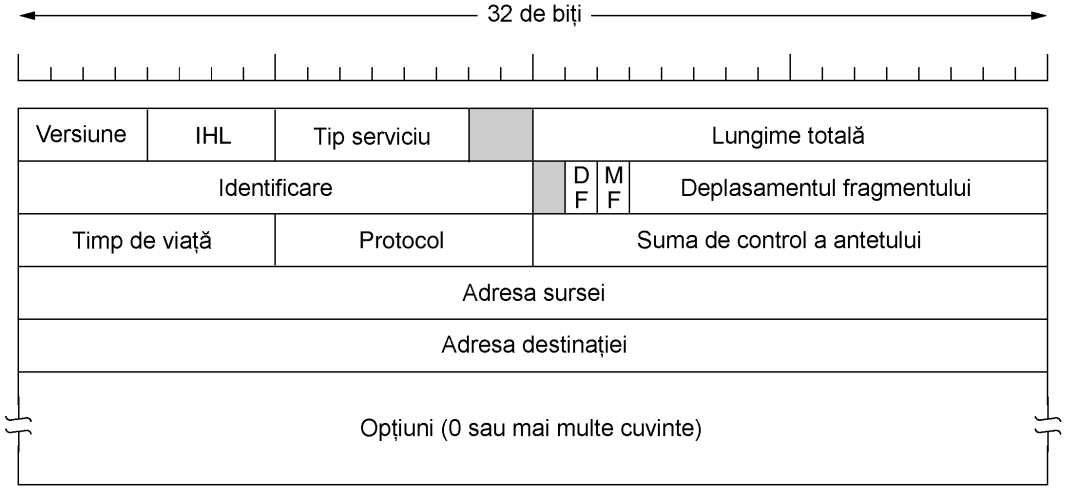


Fig. 5-53. Antetul IPv4 (protocolul Internet).

Câmpul *Versiune* memorează cărei versiuni de protocol îi aparține datagrama. Prin includerea unui câmp versiune în fiecare datagramă, devine posibil ca tranziția între versiuni să dureze ani de zile, cu unele mașini rulând vechea versiune, iar altele rulând-o pe cea nouă. La ora actuală are loc o tranziție de la IPv4 la IPv6, care deja durează de câțiva ani și în nici un caz nu s-a apropiat de final (Durand, 2001; Wilijakka, 2002; and Waddington and Chang 2002). Anumite persoane consideră chiar că nu se va întâmpla niciodată (Weiser, 2001). Referitor la numerotare, IPv5 a fost un protocol experimental de flux în timp real care nu a fost folosit pe scară largă.

Din moment ce lungimea antetului nu este constantă, un câmp din antet, *IHL*, este pus la dispoziție pentru a spune cât de lung este antetul, în cuvinte de 32 de octeți. Valoarea minimă este 5, care se aplică atunci când nu sunt prezente opțiuni. Valoarea maximă a acestui câmp de 4 biți este 15,

ceea ce limitează antetul la 60 de octeți și, astfel, câmpul de opțiuni la 40 de octeți. Pentru unele opțiuni, cum ar fi cea care înregistrează calea pe care a mers un pachet, 40 de octeți sunt mult prea puțini, făcând această opțiune nefolositoare.

Câmpul *Tip serviciu* este unul dintre puținele câmpuri care și-a schimbat sensul (oarecum) de-a lungul anilor. A fost și este în continuare menit să diferențieze diferitele clase de servicii. Sunt posibile diferite combinații de fiabilitate și viteză. Pentru vocea digitizată, livrarea rapidă are prioritate față de transmisia corectă. Pentru transferul de fișiere, transmisia fără erori este mai importantă decât transmisia rapidă.

La început, câmpul de 6 biți conținea (de la stânga la dreapta), un câmp *Precedență* de trei biți și trei indicatori, *D*, *T* și *R*. Câmpul *Precedență* reprezintă prioritatea, de la 0 (normal) la 7 (pachet de control al rețelei). Cei trei biți indicatori permiteau calculatorului gazdă să specifice ce îl afectează cel mai mult din mulțimea {Delay (Întârziere), Throughput (Productivitate), Reliability (Fiabilitate)}. În teorie, aceste câmpuri permit ruterelor să aleagă între, de exemplu, o legătură prin satelit cu o productivitate mare și o întârziere mare sau o linie dedicată cu o productivitate scăzută și o întârziere mică. În practică, ruterele curente ignoră adesea întregul câmp *Tip serviciu*.

Până la urmă, IETF a cedat și a modificat puțin câmpul pentru a-l adapta la servicii diferențiate. Șase dintre biți sunt folosiți pentru a indica căreia dintre clasele de servicii descrise mai devreme îi aparține pachetul. Aceste clase includ patru priorități de introducere în coadă, trei probabilități de respingere și clasele istorice.

Lungimea totală include totul din datagramă - atât antet cât și date. Lungimea maximă este de 65535 octeți. În prezent, această limită superioară este tolerabilă, dar în viitoarele rețele cu capacități de gigaocteți vor fi necesare datagrame mai mari. Câmpul *Identificare* este necesar pentru a permite gazdei destinație să determine cărei datagrame îi aparține un nou pachet primit. Toate fragmentele unei datagrame conțin aceeași valoare de *Identificare*.

Urmează un bit nefolosit și apoi două câmpuri de 1 bit. *DF* însemnă Don't Fragment (A nu se fragmenta). Acesta este un ordin dat ruterelor ca să nu fragmenteze datagrama pentru că destinația nu este capabilă să reasambleze piesele la loc. De exemplu, când un calculator pornește, memoria sa ROM poate cere să i se trimită o imagine de memorie ca o singură datagramă. Prin marcarea datagramei cu bitul *DF*, emițătorul știe că aceasta va ajunge într-o singură bucată, chiar dacă aceasta înseamnă că datagrama trebuie să evite o rețea cu pachete mai mici pe calea cea mai bună și să aleagă o rută suboptimală. Toate mașinile trebuie să accepte fragmente de 576 octeți sau mai mici.

MF însemnă More Fragments (mai urmează fragmente). Toate fragmentele, cu excepția ultimului, au acest bit activat. El este necesar pentru a ști când au ajuns toate fragmentele unei datagrame.

Deplasamentul fragmentului spune unde este locul fragmentului curent în cadrul datagramei. Toate fragmentele dintr-o datagramă, cu excepția ultimului, trebuie să fie un multiplu de 8 octeți - unitatea de fragmentare elementară. Din moment ce sunt prevăzuți 13 biți, există un maxim de 8192 de fragmente pe datagramă, obținându-se o lungime maximă a datagramei de 65536 octeți, cu unul mai mult decât câmpul *Lungime totală*.

Câmpul *Timp de viață* este un contor folosit pentru a limita durata de viață a pachetelor. Este prevăzut să contorizeze timpul în secunde, permițând un timp maxim de viață de 255 secunde. El trebuie să fie decrementat la fiecare salt (hop - trecere dintr-o rețea în alta) și se presupune că este decrementat de mai multe ori când stă la coadă un timp îndelungat într-un ruter. În practică, el contorizează doar salturile. Când ajunge la valoarea zero, pachetul este eliminat și se trimite înapoi la gazda sursă un pachet de avertisment. Această facilitate previne hoinăreala la infinit a datagramelor, ceea ce se poate întâmpla dacă tabelele de dirijare devin incoerente.

Când nivelul rețea a asamblat o datagramă completă, trebuie să știe ce să facă cu ea. Câmpul *Protocol* spune cărui proces de transport trebuie să o predea. TCP este o posibilitate, dar tot așa sunt și UDP și alte câteva. Numerotarea protocoalelor este globală la nivelul întregului Internet. Protocoalele și alte numere alocate erau anterior definite în RFC 1700, dar astăzi ele sunt conținute într-o bază de date on-line, care se găsește la adresa *www.iana.org*.

Suma de control a antetului verifică numai antetul. O astfel de sumă de control este utilă pentru detectarea erorilor generate de locații de memorie proaste din interiorul unui ruter. Algoritmul este de a aduna toate jumătățile de cuvinte, de 16 biți, atunci când acestea sosesc, folosind aritmetică în complement față de unu și păstrarea complementului față de unu al rezultatului. Pentru scopul acestui algoritm, se presupune că la sosire *suma de control a antetului* este zero. Acest algoritm este mai robust decât folosirea unei adunări normale. Observați că *suma de control a antetului* trebuie recalculată la fiecare salt, pentru că întotdeauna se schimbă cel puțin un câmp (câmpul *timp de viață*), dar se pot folosi trucuri pentru a accelera calculul.

Adresa sursei și *Adresa destinației* indică numărul de rețea și numărul de gazdă. Vom discuta adresele Internet în secțiunea următoare. Câmpul *Opțiuni* a fost proiectat pentru a oferi un subterfugiu care să permită versiunilor viitoare ale protocolului să includă informații care nu sunt prezente în proiectul original, pentru a permite cercetătorilor să încerce noi idei și pentru a evita alocarea unor biți din antet pentru informații folosite rar. Opțiunile sunt de lungime variabilă. Fiecare începe cu un cod de un octet care identifică opțiunea. Unele opțiuni sunt urmate de un câmp de un octet reprezentând lungimea opțiunii, urmat de unul sau mai mulți octeți de date. Câmpul *Opțiuni* este completat până la un multiplu de 4 octeți. Inițial erau definite cinci opțiuni, așa cum sunt listate în fig. 5-54, dar de atunci au mai fost adăugate și altele. Lista completă se găsește la adresa *www.iana.org/assignements/ip-parameters*.

Opțiune	Descriere
Securitate	Menționează cât de secretă este datagrama
Dirijare strictă de la sursă	Indică calea completă de parcurs
Dirijare aproximativă de la sursă	Indică o listă a ruterelor care nu trebuie sărite
Înregistrează calea	Determină fiecare ruter să-și adauge adresa IP
Amprentă de timp	Determină fiecare ruter să-și adauge adresa și o amprentă de timp.

Fig. 5-54. Unele dintre opțiunile IP.

Opțiunea *Securitate* menționează cât de secretă este informația. În teorie, un ruter militar poate folosi acest câmp pentru a menționa că nu se dorește o dirijare prin anumite țări pe care militarii le consideră a fi „băieții răi". În practică, toate ruterele îl ignoră, deci singura sa funcție practică este să ajute spionii să găsească mai ușor lucrurile de calitate.

Opțiunea *Dirijare strictă de la sursă* dă calea completă de la sursă la destinație ca o secvență de adrese IP. Datagrama este obligată să urmărească această cale precisă. Ea este deosebit de utilă pentru administratorii de sistem pentru a trimite pachete de urgență atunci când tabelele de dirijare sunt distruse sau pentru a realiza măsurători de timp.

Opțiunea *Dirijare aproximativă de la sursă* cere ca pachetul să traverseze o listă specificată de rutere și în ordinea specificată, dar este permisă trecerea prin alte rutere pe drum. În mod normal, această opțiune ar putea oferi doar câteva rutere, pentru a forța o anumită cale. De exemplu, pentru a forța un pachet de la Londra la Sydney să meargă spre vest în loc de est, această opțiune poate specifica rutere în New York, Los Angeles și Honolulu. Această opțiune este foarte utilă atunci când motive politice sau economice dictează trecerea prin anumite țări sau evitarea lor.

Opțiunea *Înregistrează calea* indică ruterelor de pe cale să-și adauge adresele IP la câmpul opțiune. Aceasta permite administratorilor de sistem să localizeze pene în algoritmii de dirijare („De ce pachetele de la Houston la Dallas trec mai întâi prin Tokio?"). Când rețeaua ARPANET a fost înființată, nici un pachet nu trecea vreodată prin mai mult de nouă rutere, deci 40 de octeți pentru opțiuni au fost destui. Așa cum s-a menționat anterior, acum dimensiunea este prea mică.

În sfârșit, opțiunea *Amprentă de timp* este similară opțiunii *Înregistrează ruta*, cu excepția faptului că, în plus față de înregistrarea adresei sale de 32 de biți, fiecare ruter înregistrează și o amprentă de timp de 32 de biți. Și această opțiune este folosită în special pentru depanarea algoritmilor de dirijare.

5.6.2 Adrese IP

Fiecare gazdă și ruter din Internet are o adresă IP, care codifică adresa sa de rețea și de gazdă. Combinația este unică: în principiu nu există două mașini cu aceeași adresă IP. Toate adresele IP sunt de 32 de biți lungime și sunt folosite în câmpurile *Adresă sursă* și *Adresă destinație* ale pachetelor IP. Este important de observat că o adresă IP nu se referă de fapt la o gazdă. Se referă de fapt la o interfață de rețea, deci dacă o gazdă este în două rețele, trebuie să folosească două adrese IP. Totuși în practică, cele mai multe gazde sunt conectate la o singură rețea și deci au o adresă IP.

Timp de mai multe decenii, adresele IP erau împărțite în cinci categorii ilustrate în fig. 5-55. Acest model de alocare a fost denumit **clase de adrese**. Nu mai este folosit, dar referințele la acest model sunt în continuare des întâlnite în literatură. Vom discuta puțin mai târziu ce a înlocuit modelul claselor de adrese.

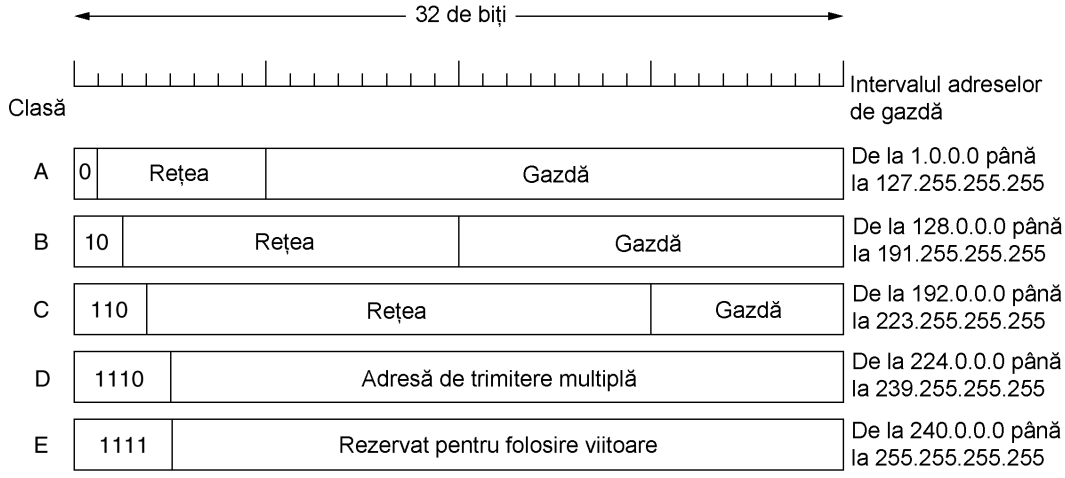


Fig. 5-55. Formatul adreselor IP.

Formatele de clasă A, B, C și D permit până la 128 rețele cu 16 milioane de gazde fiecare, 16.384 rețele cu până la 64K gazde, 2 milioane de rețele (de exemplu, LAN-uri) cu până la 256 gazde fiecare (deși unele dintre acestea sunt speciale). De asemenea este suportată și trimiterea multiplă (multicast), în care fiecare datagramă este direcționată mai multor gazde. Adresele care încep cu 1111 sunt rezervate pentru o folosire ulterioară. Peste 500 000 de rețele sunt conectate acum la Internet și numărul acestora crește în fiecare an. Pentru a evita conflictele numerele de rețea sunt atribuite de **ICANN (Internet Corporation for Assigned NAMES and Numbers** – Corporația Internet

pentru numere și nume atribuite). La rândul său, ICANN a împuternicit diverse autorități regionale să administreze părți din spațiul de adrese și acestea, la rândul lor, au împărțit adrese ISP-urilor și altor companii.

Adresele de rețea, care sunt numere de 32 de biți, sunt scrise în mod uzual în **notația zecimală cu punct**. În acest format, fiecare din cei 4 octeți este scris în zecimal, de la 0 la 255. De exemplu, adresa hexazecimală C0290614 este scrisă ca 192.41.6.20. Cea mai mică adresă IP este 0.0.0.0 și cea mai mare este 255.255.255.255.

Valorile 0 și -1 au semnificații speciale, așa cum se arată în fig. 5-56. Valoarea 0 înseamnă rețeaua curentă sau gazda curentă. Valoarea -1 este folosită ca o adresă de difuzare pentru a desemna toate gazdele din rețeaua indicată.

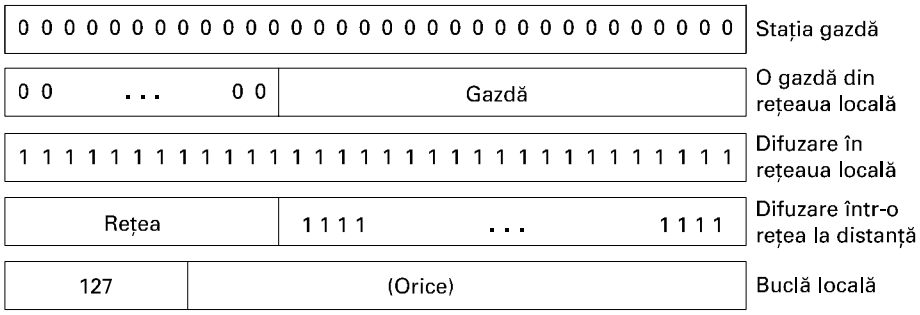

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Stația gazdă
0 0 . . . 0 0
Gazdă
O gazdă din rețeaua locală
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Difuzare în rețeaua locală
Rețea
1 1 1 1 . . . 1 1 1 1
Difuzare într-o rețea la distanță
127
(Orice)
Buclă locală

Fig. 5-56. Adrese IP speciale.

Adresa IP 0.0.0.0 este folosită de gazde atunci când sunt pornite. Adresele IP cu 0 ca număr de rețea se referă la rețeaua curentă. Aceste adrese permit ca mașinile să refere propria rețea fără a cunoaște numărul de rețea (dar ele trebuie să cunoască clasa adresei pentru a ști câte zerouri să includă). Adresele care constau numai din 1-uri permit difuzarea în rețeaua curentă, în mod uzual un LAN. Adresele cu un număr exact de rețea și numai 1-uri în câmpul gazdă permit mașinilor să trimită pachete de difuzare în LAN-uri la distanță, aflate oriunde în Internet (deși mulți administratori de sistem dezactivează această opțiune). În final, toate adresele de forma 127.xx.yy.zz sunt rezervate pentru testări în buclă locală (loopback). Pachetele trimise către această adresă nu sunt trimise prin cablu; ele sunt prelucrate local și tratate ca pachete sosite. Aceasta permite trimiterea pachetelor către rețeaua locală fără ca emițătorul să-i cunoască numărul.

Subrețele

Așa cum am văzut, toate gazdele dintr-o rețea trebuie să aibă același număr de rețea. Această proprietate a adresării IP poate crea probleme când rețeaua crește. De exemplu, să considerăm o universitate care a folosit la început o rețea de clasă B pentru calculatoarele din rețeaua Ethernet a Departamentului de Calculatoare. Un an mai târziu, departamentul de Inginerie Electrică a vrut acces la Internet, deci a cumpărat un repetor pentru a extinde Ethernetul Departamentului de Calculatoare în clădirea lor. Cu timpul, multe alte departamente au achiziționat calculatoare și limita de patru repetoare per Ethernet a fost rapid atinsă. Era nevoie de o altă organizare.

Achiziționarea altei adrese de rețea ar fi fost dificilă deoarece adresele sunt insuficiente și universitatea avea deja adrese suficiente pentru peste 60,000 de stații. Problema provine de la regula care afirmă că o singură adresă de clasă A, B sau C se referă la o singură rețea, nu la o mul-

țime de rețele. Cum tot mai multe organizații s-au lovit de această problemă, sistemul de adresare a fost puțin modificat pentru a o rezolva

Soluția este să se permită ca o rețea să fie divizată în mai multe părți pentru uz intern, dar pentru lumea exterioară să se comporte tot ca o singură rețea. În ziua de azi o rețea de campus tipică poate arăta ca în fig. 5-57, cu un ruter principal conectat la un ISP sau la rețeaua regională și numeroase rețele Ethernet împrăștiate prin campus în diferite departamente. Fiecare Ehternet are propriul ruter conectat la ruterul principal (posibil printr-un LAN coloană vertebrală, dar natura conexiunii interruter nu este relevantă aici)

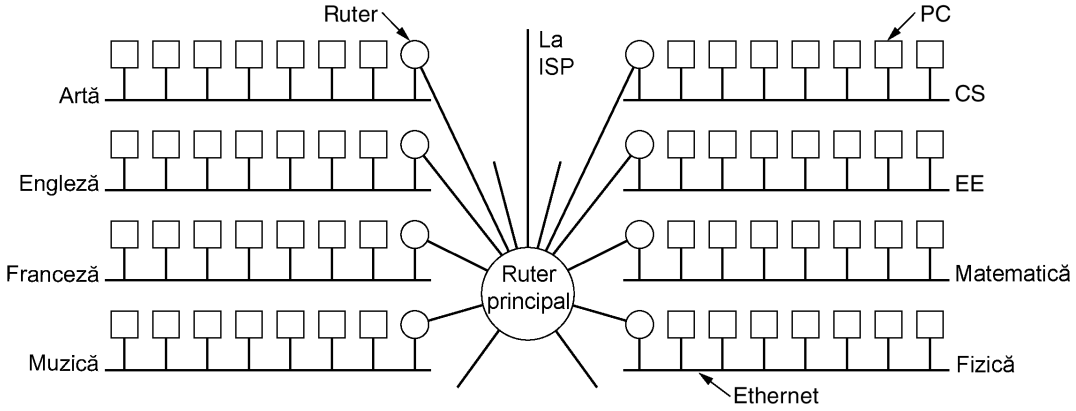


Fig. 5-57. O rețea de campus compusă din mai multe LAN-uri ale diferitelor departamente

În literatura Internet, aceste părți sunt numite **subrețele**. Așa cum am menționat în Cap. 1, această utilizare intră în conflict cu termenul „subrețea", care înseamnă mulțimea tuturor ruterelor și liniilor de comunicație dintr-o rețea. Din fericire, va fi clar din context care semnificație este atribuită. În această secțiune, și în următoarea definiția folosită va fi cea nouă.

Atunci când un pachet ajunge la ruterul principal, de unde știe acesta către ce subrețea (Ethernet) să-l trimită? O soluție ar fi să existe o tabelă cu 65,536 înregistrări în ruterul principal care să-i spună acestuia ce ruter să folosească pentru fiecare stație din campus. Această idee ar funcționa, dar ar fi nevoie de o tabelă foarte mare în ruterul principal și de multă muncă manuală de întreținere pe măsură ce stațiile ar fi adăugate, mutate sau scoase din uz.

În locul ei a fost inventată o altă soluție. În esență, în loc să existe o singură adresă de clasă B, cu 14 biți pentru numărul rețelei și 16 biți pentru gazdă, un număr de biți din numărul gazdei sunt folosiți pentru a crea un număr de subrețea. De exemplu, dacă o universitate are 35 de departamente, ar putea folosi un număr de subrețea de 6 biți și un număr de 10 biți pentru gazde, ceea ce ar permite un număr de 64 de rețele Ethernet, fiecare cu un maxim de 1022 de gazde (așa cum am menționat mai devreme, 0 și -1 nu sunt disponibile). Această împărțire ar putea fi modificată ulterior în caz că a fost o greșeală.

Pentru a se putea folosi subrețele, ruterul principal trebuie are nevoie de o **mască de subrețea,** care indică separarea dintre numărul rețea + subrețea și gazdă, așa cum este ilustrat în fig. 5-58. Măștile de subrețea sunt scrise de asemenea în notație zecimală cu punct, cu adăugarea unui caracter / (slash) urmat de numărul de biți din partea cu rețea + subrețea. Pentru exemplul din fig. 5-58, masca de subrețea poate fi scrisă ca 255.255.255.0 . O notație alternativă este /22 pentru a indica că masca subrețelei are lungimea de 22 de biți.

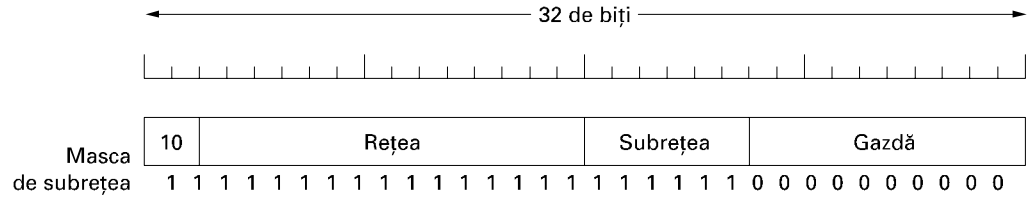


Fig. 5-58. O rețea de clasă B împărțită în 64 de subrețele

În afara rețelei, împărțirea în subrețele nu este vizibilă, astfel încât alocarea unei noi subrețele nu necesită contactarea ICANN sau schimbarea unor baze de date externe. În acest exemplu, prima subrețea poate folosi adrese IP începând de la 130.50.4.1, cea de-a doua poate începe de la 130.50.8.1, cea de-a treia poate începe de la 130.50.12.1 și așa mai departe. Pentru a se înțelege de ce numărul subrețelei crește cu patru, să observăm că adresele binare corespunzătoare sunt următoarele:

Subrețea 1 :	10000010	00110010	000001|00	00000001
Subrețea 2 :	10000010	00110010	000010|00	00000001
Subrețea 3 :	10000010	00110010	000011|00	00000001

Aici, bara verticală (|) arată granița dintre numărul subrețelei și numărul gazdei. La stânga se găsește numărul de 6 biți al subrețelei, la dreapta numărul de 10 biți al gazdei.

Pentru a vedea cum funcționează subrețelele, este necesar să explicăm cum sunt prelucrate pachetele IP într-un ruter. Fiecare ruter are o tabelă ce memorează un număr de adrese IP de forma (rețea, 0) și un număr de adrese IP de forma (această-rețea, gazdă). Primul tip indică cum se ajunge la rețelele aflate la distanță. Al doilea tip spune cum se ajunge la gazdele locale. Fiecărei tabele îi este asociată interfața de rețea care se folosește pentru a ajunge la destinație și alte câteva informații.

Când sosește un pachet IP, adresa destinație este căutată în tabela de dirijare. Dacă pachetul este pentru o rețea aflată la distanță, el este trimis ruterului următor prin interfața specificată în tabelă. Dacă este o gazdă locală (de exemplu în LAN-ul ruterului), pachetul este trimis direct către destinație. Dacă rețeaua nu este prezentă, pachetul este trimis unui ruter implicit care are tabele mai extinse. Acest algoritm înseamnă că fiecare ruter trebuie să memoreze numai rețele și gazde, nu perechi (rețea, gazdă), reducând considerabil dimensiunea tabelelor de dirijare.

Când este introdusă împărțirea în subrețele, tabelele de dirijare sunt schimbate, adăugând intrări de forma (această-rețea, subrețea, 0) și (această-rețea, această-subrețea, gazdă). Astfel un ruter din subrețeaua *k* știe cum să ajungă la toate celelalte subrețele și, de asemenea, cum să ajungă la toate gazdele din subrețeaua *k*. El nu trebuie să cunoască detalii referitoare la gazde din alte subrețele. De fapt, tot ceea ce trebuie schimbat este de a impune fiecărui ruter să facă un ȘI logic cu **masca de subrețea** a rețelei pentru a scăpa de numărul de gazdă și a căuta adresa rezultată în tabelele sale (după ce determină cărei clase de rețele aparține). De exemplu, asupra unui pachet adresat către 130.50.15.6 care ajunge la ruterul principal se face un ȘI logic cu masca de subrețea 255.255.252.0/22 pentru a obține adresa 130.50.12.0. Această adresă este căutată în tabelele de dirijare pentru a se găsi cum se ajunge la gazdele din subrețeaua 3. Ruterul din subrețeaua 5 este astfel ușurat de munca de a memora toate adresele de nivel legătură de date ale altor gazde decât cele din subrețeaua 5. Împărțirea în subrețele reduce astfel spațiul tabelelor de dirijare prin crearea unei ierarhii pe trei niveluri, alcătuită din rețea, subrețea și gazdă.

CIDR - Dirijarea fără clase între domenii

IP este folosit intens de câteva decenii. A funcționat extrem de bine, așa cum a fost demonstrat de creșterea exponențială a Internet-ului. Din nefericire, IP devine rapid o victimă a propriei popularități: își epuizează adresele. Acest dezastru fantomatic a generat foarte multe discuții și controverse în cadrul comunității Internet referitor la cum să fie tratat. În această secțiune vom descrie atât problema cât și câteva soluții propuse.

Prin 1987, câțiva vizionari au prezis că într-o zi Internet-ul poate crește până la 100.000 de rețele. Cei mai mulți experți s-au exprimat cu dispreț că aceasta va fi peste decenii, dacă va fi vreodată. Cea de-a 100.000-a rețea a fost conectată în 1996. Problema, expusă anterior, este că Internet-ul își epuizează rapid adresele IP. În principiu, există peste 2 miliarde de adrese, dar practica organizării spațiului de adrese în clase (vezi fig. 5-55) irosește milioane din acestea. În particular, adevărații vinovați sunt de rețelele de clasă B. Pentru cele mai multe organizații, o adresă de clasă A, cu 16 milioane de adrese este prea mare, iar o rețea de clasă C, cu 256 de adrese, este prea mică. O rețea de clasă B, cu 65.536 adrese, este numai bună. În folclorul Internet, această situație este cunoscută ca **problema celor trei urși** (ca din *Goldilocks and the Three Bears*).

În realitate, o adresă de clasă B este mult prea mare pentru cele mai multe organizații. Studiile au arătat că mai mult de jumătate din toate rețelele de clasă B au mai puțin de 50 de gazde. O rețea de clasă C ar fi fost suficientă, dar fără îndoială că fiecare organizație care a cerut o adresă de clasă B a crezut că într-o zi ar putea depăși câmpul gazdă de 8 biți. Privind retrospectiv, ar fi fost mai bine să fi avut rețele de clasă C care să folosească 10 biți în loc de opt pentru numărul de gazdă, permițând 1022 gazde pentru o rețea. În acest caz, cele mai multe organizații s-ar fi decis, probabil, pentru o rețea de clasă C și ar fi existat jumătate de milion dintre acestea (comparativ cu doar 16.384 rețele de clasă B).

Este greu să fie învinuiți proiectanții Internetului pentru că nu au pus la dispoziție mai multe adrese de clasă B (și mai mici). În momentul în care s-a luat decizia să fie create cele trei clase, Internetul era o rețea de cercetare care conecta marile universități din SUA (plus un număr mic de companii și sit-uri militare care făceau cercetări în domeniul rețelelor). Pe atunci nimeni nu a perceput Internetul ca fiind un sistem de comunicație în masă care va rivaliza cu rețeaua telefonică. Fără îndoială că în acel moment cineva a spus: „SUA are aproximativ 2000 de universități și colegii. Chiar dacă toate se conectează la Internet și se mai conectează și multe universități din alte țări, nu o să ajungem niciodată la 16.000 pentru că nu există atâtea universități în întreaga lume. În plus faptul că numărul gazdei este un număr întreg de octeți, mărește viteza de prelucrare a pachetelor."

Totuși, dacă s-ar fi alocat 20 de biți numărului de rețea pentru rețele de clasă B, ar fi apărut o altă problemă: explozia tabelelor de dirijare. Din punctul de vedere al ruterelor, spațiul de adrese IP este o ierarhie pe două niveluri, cu numere de rețea și numere de gazde. Ruterele nu trebuie să știe despre toate gazdele, dar ele trebuie să știe despre toate rețelele. Dacă ar fi fost în folosință jumătate de milion de rețele de clasă C, fiecare ruter din întregul Internet ar fi necesitat o tabelă cu o jumătate de milion de intrări, una pentru fiecare rețea, spunând care linie se folosește pentru a ajunge la respectiva rețea, împreună cu alte informații.

Memorarea fizică efectivă a tabelelor cu jumătate de milion de intrări este probabil realizabilă, deși costisitoare pentru ruterele critice care trebuie să mențină tabelele în RAM static pe plăcile I/O. O problemă mai serioasă este reprezentată de creșterea complexității diferiților algoritmi referitori la gestiunea tabelelor, care este mai rapidă decât liniară. Și mai rău, o mare parte din programele și firmware-ul ruterelor existente a fost proiectat pe vremea când Internet-ul avea 1000 de rețele co-

nectate și 10.000 de rețele păreau la depărtare de decenii. Deciziile de proiectare făcute atunci sunt departe de a fi optime acum.

În plus, diferiți algoritmi de dirijare necesită ca fiecare ruter să-și transmită tabelele periodic (de exemplu protocolul vectorilor distanță). Cu cât tabelele sunt mai mari, cu atât este mai probabil ca anumite părți să se piardă pe drum, ducând la date incomplete la celălalt capăt și, posibil, la instabilități de dirijare.

Problema tabelelor de dirijare ar fi putut fi rezolvată prin adoptarea unei ierarhii mai adânci. De exemplu, ar fi mers dacă s-ar fi impus ca fiecare adresă să conțină un câmp țară, județ/provincie, oraș, rețea și gazdă. Atunci fiecare ruter ar fi trebuit să știe doar cum să ajungă la fiecare țară, la județele sau provinciile din țara sa, orașele din propriul județ sau provincie și rețelele din orașul său. Din nefericire, această soluție ar necesita mai mult de 32 de biți pentru adrese IP și ar folosi adresele ineficient (Liechtenstein ar fi avut tot atâția biți ca Statele Unite).

Pe scurt, cele mai multe soluții rezolvă o problemă, dar creează o alta. Soluția care a fost implementată acum și care a dat Internet-ului un pic de spațiu de manevră este **CIDR** (**Classless InterDomain Routing** - Dirijarea fără clase între domenii). Ideea de la baza CIDR, descrisă în RFC 1519, este de a aloca adresele IP rămase, în blocuri de dimensiune variabilă, fără a se ține cont de clase. Dacă un sit are nevoie, să zicem, de 2000 de adrese, îi este dat un bloc de 2048 adrese la o graniță de 2048 de octeți.

Renunțarea la clase face rutarea mai complicată. În vechiul sistem cu clase, rutarea se efectua în felul următor. Atunci când un pachet ajungea la un ruter, o copie a adresei IP era deplasată la dreapta cu 28 de biți pentru a obține un număr de clasă de 4 biți. O ramificație cu 16 căi sorta pachetele în A, B, C și D (dacă era suportat), cu 8 cazuri pentru clasa A, patru cazuri pentru clasa B, două cazuri pentru clasa C și câte unul pentru D și E. Programul pentru fiecare clasă masca numărul de rețea de 8, 16 sau 24 de biți și îl alinia la dreapta într-un cuvânt de 32 de biți. Numărul de rețea era apoi căutat în tabela pentru A, B sau C, de obicei indexat pentru rețelele A și B și folosind dispersia (hashing) pentru rețelele C. Odată intrarea găsită, se putea găsi linia de ieșire și pachetul era retransmis.

Cu CIDR, acest algoritm simplu nu mai funcționează. În loc de aceasta, fiecare intrare din tabela de rutare este extinsă cu o mască de 32 de biți. Din acest motiv, acum există o singură tabelă de rutare pentru toate rețelele, constituită dintr-un un vector de triplete (adresă IP, mască subrețea, linie de ieșire). Când sosește un pachet IP, mai întâi se extrage adresa IP a destinației. Apoi (conceptual) tabela de rutare este scanată intrare cu intrare, mascând adresa destinație și comparând cu intrarea din tabelă, în căutarea unei potriviri. Este posibil ca mai multe intrări (cu măști de subrețea de lungimi diferite) să se potrivească, caz în care este folosită cea mai lungă mască. Astfel, dacă există potrivire pentru o mască /20 și pentru o mască /24, este folosită intrarea cu /24.

Pentru a accelera procesul de găsire a adreselor au fost imaginați algoritmi complecși (Ruiz-Sanchez et al., 2001). Ruterele comerciale folosesc chip-uri VLSI proprietare cu acești algoritmi implementați în hardware.

Pentru a face algoritmul de retransmitere mai ușor de înțeles, să considerăm un exemplu în care sunt disponibile milioane de adrese, începând de la 194.24.0.0. Să presupunem că Universitatea Cambridge are nevoie de 2048 de adrese și are atribuite adresele de la 194.24.0.0 până la 194.24.7.255, împreună cu masca 255.255.248.0. Apoi, Universitatea Oxford cere 4096 de adrese. Deoarece un bloc de 4096 de adrese trebuie să fie aliniat la o frontieră de 4096 octeți, acestea nu pot fi numerotate începând de la 194.8.0.0. În loc, se primesc adrese de la 194.24.16.0 până la 194.24.31.255, împreună cu o mască de 255.255.240.0. Acum Universitatea din Edinburgh cere

1024 de adrese și îi sunt atribuite adresele de la 194.24.8.0 până la 194.24.11.255 și masca 255.255.252.0. Alocările sunt rezumate în fig. 5-59.

Universitatea	Prima adresă	Ultima adresă	Număr adrese	Notație
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(disponibil)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Fig. 5-59. Un set de atribuiri de adrese IP.

Tabelele de dirijare din toată lumea sunt acum actualizate cu cele trei intrări atribuite. Fiecare intrare conține o adresă de bază și o mască de subrețea. Aceste intrări (în binar) sunt:

	Adresă	Mască
C	11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E	11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
O	11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

Să vedem acum ce se întâmplă când sosește un pachet adresat pentru 194.24.17.4, care este reprezentat ca următorul șir de 32 de biți :

11000010 00011000 00010001 00000100

Mai întâi, se face un ȘI logic cu masca de la Cambridge pentru a obține:

11000010 00011000 00010000 00000000

Această valoare nu se potrivește cu adresa de bază de la Cambridge, așa că adresei originale i se aplică un ȘI logic cu masca de la Edinburgh pentru a se obține:

11000010 00011000 00010000 00000000

Această valoare nu se potrivește cu adresa de bază de la Edinburgh, așa că se încearcă în continuare Oxford, obținându-se:

11000010 00011000 00010000 00000000

Această valoare se potrivește cu adresa de bază de la Oxford. Dacă în restul tabelei nu sunt găsite potriviri mai lungi, atunci este folosită intrarea pentru Oxford și pachetul va fi trimis pe linia corespunzătoare.

Acum să privim cele trei universități din punctul de vedere al unui ruter din Omaha, Nebraska, care are doar patru linii de ieșire: Minneapolis, New York, Dallas și Denver. Atunci când software-ul ruterului primește cele trei noi intrări, realizează că le poate combina pe toate trei într-o singură **intrare agregată** (eng,: **aggregate entry**) 194.24.0.0/19 cu adresa binară și o masca de subrețea următoare :

11000010 00000000 00000000 00000000	11111111 11111111 11100000 00000000

Această intrare trimite toate pachetele destinate uneia dintre cele trei universități la New York. Reunind cele trei intrări ruterul din Omaha și-a redus mărimea tabelei de rutare cu doi.

Dacă New York-ul are o singură conexiune cu Londra pentru tot traficul spre Marea Britanie, poate folosi de asemenea o intrare agregată. Totuși, dacă are două conexiuni separate pentru Londra și Edinburgh, atunci trebuie să aibă trei intrări.

Ca o notă finală despre acest exemplu, intrarea agregată a ruterului din Omaha va trimite și pachetele către adresele nealocate tot spre New York. Cât timp adresele sunt cu adevărat nealocate, aceasta nu contează pentru că așa ceva nu ar trebui să se întâmple. Totuși, dacă mai târziu sunt alocate unei companii din California, pentru a le trata va fi nevoie de o înregistrare separată, 194.24.12.0/22.

NAT – Translatarea adreselor de rețea

Adresele IP sunt insuficiente. Un ISP ar putea avea o adresă /16 (anterior de clasă B) oferindu-i 65.534 numere de stații. Dacă are mai mulți clienți, are o problemă. Pentru utilizatorii casnici cu conexiuni pe linie telefonică (eng.: dial-up), o soluție ar fi să se aloce dinamic o adresă IP fiecărui calculator în momentul în care sună și se conectează și să o dealoce în momentul în care se termină sesiunea. În acest fel o singură adresă /16 poate fi folosită pentru 65.534 utilizatori activi, ceea ce este probabil suficient pentru un ISP cu mai multe sute de mii de utilizatori. În momentul în care sesiunea se termină, adresa IP este alocată altui calculator care sună. Deși această strategie funcționează bine pentru un ISP cu un număr moderat de utilizatori casnici, eșuează pentru ISP-uri care deservesc preponderent companii.

Problema este că o companie se așteaptă să fie on-line în mod continuu în timpul orelor de program. Atât companiile mici, cum ar fi o companie de turism compusă din trei persoane, cât și mari corporații au mai multe calculatoare conectate de un LAN. Unele sunt calculatoarele personale ale angajaților; altele pot fi servere Web. În general în LAN există un ruter care este conectat cu ISP-ul printr-o linie închiriată pentru a avea conexiune continuă. Această situație impune ca fiecărui calculator să îi fie asociat un IP propriu pe parcursul întregii zile. De fapt, numărul total al calculatoarelor companiilor care sunt clienți ai ISP-ului nu poate depăși numărul de adrese IP pe care le posedă acesta. Pentru o adresă /16, aceasta limitează numărul total de calculatoare la 65.534. Pentru un ISP cu zeci de mii de companii cliente, această limită va fi repede depășită.

Pentru a agrava situația, din ce în ce mai mulți utilizatori se abonează de acasă la ADSL sau Internet prin cablu. Două dintre facilitățile oferite de aceste servicii sunt (1) utilizatorul primește o adresă IP permanentă și (2) nu există taxă de conectare (doar o taxă lunară), așa că mulți utilizatori ai ADSL-ului și ai Internetului prin cablu rămân conectați permanent. Aceasta contribuie la insuficiența adreselor IP. Alocarea dinamică a adreselor IP așa cum se face cu utilizatorii dial-up nu este utilă pentru că numărul adreselor IP folosite la un moment dat poate fi de multe ori mai mare decât numărul adreselor deținute de ISP.

Și pentru a complica și mai mult lucrurile, mulți utilizatori de ADSL și Internet prin cablu au două sau mai multe calculatoare acasă, deseori câte unul pentru fiecare membru al familiei, și toți vor să fie online tot timpul folosind singura adresă IP care le-a fost alocată de către ISP. Soluția este să se conecteze toate PC-urile printr-un LAN și să se pună un ruter. Din punctul de vedere al ISP-ului, familia este acum ca o companie cu câteva calculatoare. Bine-ați venit la Jones, Inc.

Problema epuizării adreselor IP nu este o problemă teoretică care ar putea apărea cândva în viitorul îndepărtat. A apărut aici și acum. Soluția pe termen lung este ca tot Internetul să migreze la IPv6, care are adrese de 128 de biți. Tranziția se desfășoară încet, dar vor trece ani până la finalizarea procesului. În consecință, anumite persoane au considerat că este nevoie de o rezolvare rapidă pe termen scurt. Rezolvarea a venit în forma NAT (Network Address Translation –

Translatarea adreselor de rețea), care este descrisă în RFC 3022 și pe care o vom rezuma mai jos. Pentru informații suplimentare consultați (Dutcher, 2001).

Ideea de bază a NAT-ului este de a aloca fiecărei companii o singură adresă IP (sau cel mult un număr mic de adrese) pentru traficul Internet. În interiorul companiei, fiecare calculator primește o adresă IP unică, care este folosită pentru traficul intern. Totuși, atunci când un pachet părăsește compania și se duce la ISP, are loc o translatare de adresă. Pentru a face posibil acest lucru, au fost declarate ca fiind private trei intervale de adrese IP. Companiile le pot folosi intern așa cum doresc. Singura regulă este ca nici un pachet conținând aceste adrese să nu apară pe Internet. Cele trei intervale rezervate sunt :

10.0.0.0	- 10.255.255.255/8	(16.777.216 gazde)
172.16.0.0	- 172.31.255.255/12	(1 .048.576 gazde)
192.168.0.0	- 192.168.255.255/16	(65.536 gazde)

Primul interval pune la dispoziție 16.777.216 adrese (cu excepția adreselor 0 și –1, ca de obicei) și este alegerea obișnuită a majorității companiilor, chiar dacă nu au nevoie de așa de multe adrese.

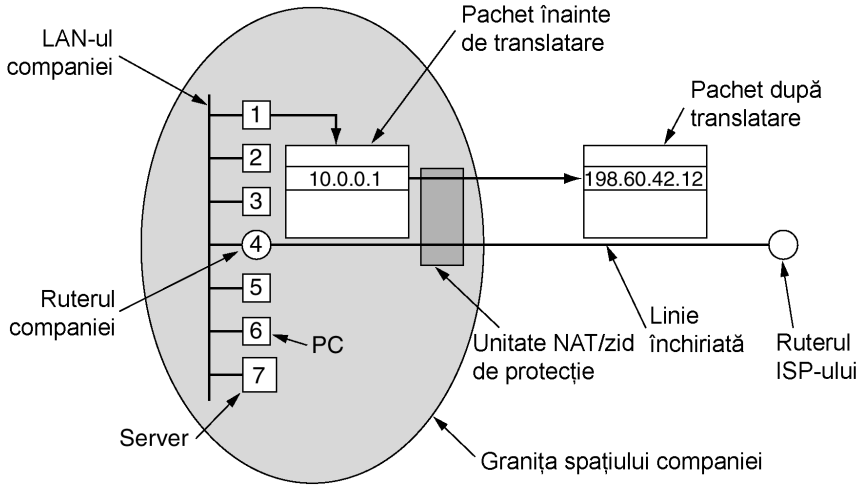


Fig. 5-60. Amplasarea și funcționarea unei unități NAT.

Funcționarea NAT este ilustrată în fig. 5-60. În interiorul companiei fiecare mașină are o adresă unică de forma 10.x.y.z. Totuși, când un pachet părăsește compania, trece printr-o **unitate NAT** (eng.: **NAT box**)care convertește adresa IP internă, 10.0.0.1 în figură, la adresa IP reală a companiei, 198.60.42.12, în acest exemplu. Unitatea NAT este deseori combinată într-un singur dispozitiv cu un zid de protecție (eng.: firewall), care asigură securitatea controlând cu grijă ce intră și iese din companie. Vom studia zidurile de protecție în Cap. 8. De asemenea este posibilă integrarea unității NAT în ruterul companiei.

Până acum am trecut cu vederea un detaliu: atunci când vine răspunsul (de exemplu de la un server Web), este adresat 198.60.42.12, deci de unde știe unitatea NAT cu care adresă să o înlocuiască pe aceasta? Aici este problema NAT-ului. Dacă ar fi existat un câmp liber în antetul IP, acel câmp ar fi putut fi folosit pentru a memora cine a fost adevăratul emițător, dar numai 1 bit este încă nefolosit. În principiu, o nouă opțiune ar putea fi creată pentru a reține adevărata adresă a

sursei, dar acest lucru ar necesita schimbarea codului din protocolul IP de pe toate stațiile din întregul Internet pentru a putea prelucra noua opțiune. Aceasta nu este o alternativă promițătoare pentru o rezolvare rapidă.

În cele ce urmează se prezintă ceea ce s-a întâmplat cu adevărat. Proiectanții NAT au observat că marea majoritate a pachetelor IP aveau conținut TCP sau UDP. Atunci când vom studia protocoalele TCP și UDP în Cap. 6, vom vedea că ambele au antete conțin un port sursă și un port destinație. În continuare vom discuta doar despre porturi TCP, dar exact aceleași lucruri se aplică și la UDP. Porturile sunt numere întregi pe 16 biți care indică unde începe și unde se termină o conexiune TCP. Aceste câmpuri pun la dispoziție câmpul necesar funcționării NAT.

Atunci când un proces dorește să stabilească o conexiune TCP cu un proces de la distanță, se atașează unui port TCP nefolosit de pe mașina sa. Acesta se numește **portul sursă** și spune codului TCP unde să trimită pachetele care vin și aparțin acestei conexiuni. Procesul furnizează și un **port destinație** pentru a spune cui să trimită pachetele în cealaltă parte. Porturile 0-1023 sunt rezervate pentru servicii bine cunoscute. De exemplu portul 80 este folosit de serverele Web, astfel încât clienții să le poată localiza. Fiecare mesaj TCP care pleacă conține atât un port sursă cât și un port destinație. Cele două porturi permit identificarea proceselor care folosesc conexiunea la cele două capete.

S-ar putea ca o analogie să clarifice mai mult utilizarea porturilor. Imaginați-vă o companie cu un singur număr de telefon principal. Atunci când cineva sună la acest număr, vorbește cu un operator care întreabă ce interior dorește și apoi face legătura la acel interior. Numărul principal este analog cu adresa IP a companiei iar interioarele de la ambele capete sunt analoage cu porturile. Porturile reprezintă 16 biți suplimentari de adresare identificând procesul care primește un pachet sosit.

Folosind câmpul *Port sursă* rezolvăm problema de corespondență. De fiecare dată când un pachet pleacă, el intră în unitatea NAT și adresa sursă 10.x.y.z este înlocuită cu adresa reală a companiei. În plus, câmpul TCP *Port sursă* este înlocuit cu un index în tabela de translatare a unității NAT, care are 65.536 intrări. Această tabelă conține adresa IP inițială și portul inițial. În final, sunt recalculate și inserate în pachet sumele de control ale antetelor IP și TCP. Câmpul *Port sursă* trebuie înlocuit pentru că s-ar putea întâmpla, de exemplu, ca stațiile 10.0.0.1 și 10.0.0.2 să aibă amândouă conexiuni care să folosească portul 5000, deci câmpul *Port sursă* nu este suficient pentru a identifica procesul emițător.

Atunci când la unitatea NAT sosește un pachet de la ISP, *Portul sursă* din antetul TCP este extras și folosit ca index în tabela de corespondență a unității NAT. Din intrarea localizată sunt extrase și inserate în pachet adresa IP internă și *Portul sursă* TCP inițial. Apoi sunt recalculate sumele de control TCP și IP și inserate în pachet. După aceea pachetul este transferat ruterului companiei pentru transmitere normală folosind adresa 10.x.y.z.

NAT poate fi folosit pentru rezolva insuficiența adreselor IP pentru utilizatori de ADSL și Internet prin cablu. Atunci când un ISP alocă fiecărui utilizator o adresă, folosește adrese 10.x.y.z. Atunci când pachete de la stațiile utilizatorilor ies din ISP și intră în Internet trec printr-o cutie NAT care le translatează în adresele Internet adevărate ale ISP-ului. La întoarcere pachetele trec prin maparea inversă. Din această perspectivă, pentru restul Internetului, ISP-ul și utilizatorii săi ADSL/cablu arată la fel ca o mare companie.

Deși această schemă rezolvă într-un fel problema, multe persoane din comunitatea IP o consideră un monstru pe fața pământului. Rezumate succint, iată câteva dintre obiecții. În primul rând, NAT violează modelul arhitectural al IP-ului, care afirmă că fiecare adresă IP identifică unic o sin-

gură mașină din lume. Întreaga structură software a Internetului este construită pornind de la acest fapt. Cu NAT, mii de mașini pot folosi (și folosesc) adresa 10.0.0.1.

În al doilea rând, NAT schimbă natura Internetului de la o rețea fără conexiuni la un fel de rețea cu conexiuni. Problema este că o unitate NAT trebuie să mențină informații (corespondențe) pentru fiecare conexiune care trece prin ea. Faptul că rețeaua păstrează starea conexiunilor este o proprietate a rețelelor orientate pe conexiune, nu a celor neorientate pe conexiune. Dacă o unitate NAT se defectează și tabela de corespondență este pierdută, toate conexiunile TCP sunt distruse. În absența NAT, căderea ruterelor nu are nici un efect asupra TCP. Procesul care transmite ajunge la limita de timp în câteva secunde și retransmite toate pachetele neconfirmate. Cu NAT, Internetul devine la fel de vulnerabil ca o rețea cu comutare de circuite.

În al treilea rând, NAT încalcă cea mai fundamentală regulă a protocoalelor pe niveluri: nivelul *k* nu poate face nici un fel de presupuneri referitor la ceea ce a pus nivelul *k+1* în câmpul de informație utilă. Principiul de bază este să se păstreze niveluri independente. Dacă mai târziu TCP este înlocuit de TCP-2, cu un antet diferit (de exemplu porturi pe 32 de biți), NAT va eșua. Ideea protocoalelor pe niveluri este de a asigura că modificările la un nivel nu necesită modificări la celelalte niveluri. NAT distruge această independență.

În al patrulea rând, procesele din Internet nu sunt obligate să folosească TCP sau UDP. Dacă un utilizator de pe mașina *A* se decide să folosesc un nou protocol de transport pentru a comunica cu un utilizator de pe mașina *B* (de exemplu, pentru o aplicație multimedia), introducerea unei unități NAT va determina eșecul aplicației, deoarece cutia NAT nu va fi în stare să localizeze corect câmpul TCP *Port sursă*.

În al cincilea rând, anumite aplicații introduc adrese IP în corpul mesajului. Receptorul extrage aceste adrese și le folosește. De vreme ce NAT nu știe nimic despre aceste adrese, nu le poate înlocui, deci orice încercare de a le folosi de către cealaltă parte va eșua. **FTP**, standardul **File Transfer Protocol** (rom.: protocol de transfer de fișiere) funcționează în acest mod și poate eșua în prezența NAT dacă nu se iau măsuri speciale. În mod similar protocolul pentru telefonie Internet H.323 (care va fi studiat în Cap. 8) are această proprietate și nu va funcționa în prezența NAT. Ar fi posibil să se modifice NAT-ul pentru a funcționa cu H.323, dar modificarea codului unității NAT de fiecare dată când apare o aplicație nouă nu este o idee bună.

În al șaselea rând, deoarece câmpul TCP *Port sursă* are 16 biți, o adresă IP poate fi pusă în corespondență cu cel mult 65.536 mașini. De fapt acest număr este puțin mai mic, deoarece primele 4096 porturi sunt rezervate pentru utilizări speciale. Totuși dacă sunt disponibile mai multe adrese IP fiecare poate trata 61.440 mașini.

Acestea și alte probleme ale NAT sunt discutate în RFC 2993. În general, opozanții NAT spun că rezolvând problema insuficienței adreselor IP cu o soluție temporară și urâtă, presiunea pentru a implementa soluția reală, adică tranziția la IPv6, este redusă și acesta este un lucru rău.

5.5.4 Protocoale de control în Internet

Pe lângă IP, care este folosit pentru transferul de date, Internet-ul are câteva protocoale de control la nivelul rețea, incluzând ICMP, ARP, RARP, BOOTP și DHCP. În această secțiune vom arunca o privire asupra fiecăruia dintre ele.

Protocolul mesajelor de control din Internet

Operarea Internet-ului este strâns monitorizată de către rutere. Atunci când se întâmplă ceva neobișnuit, evenimentul este raportat prin **ICMP** (**Internet Control Message Protocol** - protocolul mesajelor de control din Internet), care este folosit și pentru testarea Internet-ului. Sunt definite aproape o duzină de tipuri de mesaje ICMP. Cele mai importante sunt enumerate în fig. 5-61. Fiecare tip de mesaj ICMP este încapsulat într-un pachet IP.

Tipul mesajului	Descriere
Destinație inaccesibilă	Pachetul nu poate fi livrat
Timp depășit	Câmpul timp de viață a ajuns la 0
Problemă de parametru	Câmp invalid în antet
Oprire sursă	Pachet de șoc
Redirectare	Învață un ruter despre geografie
Cerere de ecou	Întreabă o mașină dacă este activă
Răspuns ecou	Da, sunt activă
Cerere de amprentă de timp	La fel ca cererea de ecou, dar cu amprentă de timp
Răspuns cu amprentă de timp	La fel ca răspunsul ecou, dar cu amprentă de timp

Fig. 5-61. Tipurile principale de mesaje ICMP.

Mesajul DESTINAȚIE INACCESIBILĂ (DESTINATION UNREACHABLE) este folosit atunci când subrețeaua sau un ruter nu pot localiza destinația, sau un pachet cu bitul DF nu poate fi livrat deoarece o rețea cu „pachete mici" îi stă în cale.

Mesajul TIMP DEPĂȘIT (TIME EXCEEDED) este trimis când un pachet este eliminat datorită ajungerii contorului său la zero. Acest mesaj este un simptom al buclării pachetelor, al unei enorme congestii sau al stabilirii unor valori prea mici pentru ceas.

Mesajul PROBLEMĂ DE PARAMETRU (PARAMETER PROBLEM) indică detectarea unei valori nepermise într-un câmp din antet. Această problemă indică o eroare în programele IP ale gazdei emițătoare sau eventual în programele unui ruter tranzitat.

Mesajul OPRIRE SURSĂ (SOURCE QUENCH) a fost folosit pe vremuri pentru a limita traficul gazdelor care trimiteau prea multe pachete. Când o gazdă primea acest mesaj, era de așteptat să încetinească ritmul de transmisie. Este folosit arareori, deoarece când apare congestie, aceste pachete au tendința de a turna mai mult gaz pe foc. Controlul congestiei în Internet este făcut acum pe larg la nivelul transport și va fi studiat în detaliu în Cap. 6.

Mesajul REDIRECTARE (REDIRECT) este folosit atunci când un ruter observă că un pachet pare a fi dirijat greșit. Este folosit de ruter pentru a spune gazdei emițătoare despre eroarea probabilă.

Mesajele CERERE ECOU (ECHO REQUEST) și RĂSPUNS ECOU (ECHO REPLY) sunt folosite pentru a vedea dacă o anumită destinație este accesibilă și activă. Este de așteptat ca la recepția mesajului ECOU, destinația să răspundă printr-un mesaj RĂSPUNS ECOU. Mesajele CERERE AMPRENTĂ DE TIMP (TIMESTAMP REQUEST) și RĂSPUNS AMPRENTĂ DE TIMP (TIMESTAMP REPLY) sunt similare, cu excepția faptului că în răspuns sunt înregistrate timpul de sosire a mesajului și de plecare a răspunsului. Această facilitate este folosită pentru a măsura performanțele rețelei.

În plus față de aceste mesaje, au fost definite și altele. Lista se află on-line la adresa www.iana.org/assignements/icmp-parameters.

Protocolul de rezoluție a adresei: ARP

Deși fiecare mașină din Internet are una sau mai multe adrese IP, acestea nu pot fi folosite de fapt pentru trimiterea pachetelor deoarece hardware-ul nivelului legăturii de date nu înțelege adresele Internet. Actualmente, cele mai multe gazde sunt atașate la un LAN printr-o placă de interfață care înțelege numai adresele LAN. De exemplu, fiecare placă Ethernet fabricată până acum vine cu o adresă Ethernet de 48 biți. Fabricanții plăcilor Ethernet cer un spațiu de adrese de la o autoritate centrală pentru a se asigura că nu există două plăci cu aceeași adresă (pentru a evita conflictele care ar apărea dacă cele două plăci ar fi în același LAN). Plăcile trimit și primesc cadre pe baza adresei Ethernet de 48 biți. Ele nu știu absolut nimic despre adresele IP pe 32 de biți.

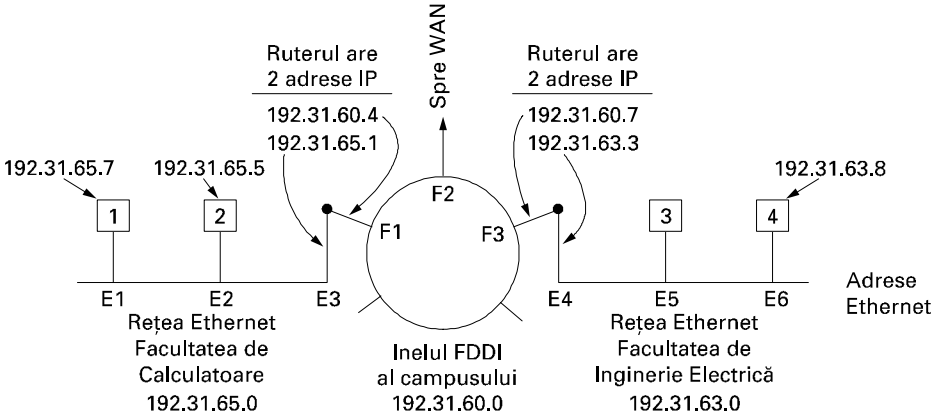


Fig. 5-62. Trei rețele /24 interconectate: două rețele Ethernet și un inel FDDI.

Se pune atunci întrebarea: Cum sunt transformate adresele IP în adrese la nivelul legăturii de date, ca de exemplu Ethernet? Pentru a explica care este funcționarea, vom folosi exemplul din fig. 5-62, în care este ilustrată o universitate mică ce are câteva rețele de clasă C (denumită acum /24). Avem două rețele Ethernet, una în facultatea de Calculatoare, cu adresa IP 192.31.65.0 și una în facultatea de Inginerie Electrică, cu adresa IP 192.31.63.0. Acestea sunt conectate printr-un inel FDDI la nivelul campusului, care are adresa IP 192.31.60.0. Fiecare mașină dintr-o rețea Ethernet are o adresă Ethernet unică, etichetată de la E1 la E6, iar fiecare mașină de pe inelul FDDI are o adresă FDDI, etichetată de la F1 la F3.

Să începem prin a vedea cum trimite un utilizator de pe gazda 1 un pachet unui utilizator de pe gazda 2. Presupunem că expeditorul știe numele destinatarului, ceva de genul *ary@eagle.cs.uni.edu*. Primul pas este aflarea adresei IP a gazdei 2, cunoscută ca *eagle.cs.uni.edu*. Această căutare este făcută de sistemul numelor de domenii (DNS), pe care îl vom studia în Cap. 7. Pentru moment, vom presupune că DNS-ul întoarce adresa IP a gazdei 2 (192.31.65.5).

Programele de la nivelurile superioare ale gazdei 1 construiesc un pachet cu 192.31.65.5 în câmpul *adresa destinatarului,* pachet care este trimis programelor IP pentru a-l transmite. Programele IP se uită la adresă și văd că destinatarul se află în propria rețea, dar au nevoie de un mijloc prin care să determine adresa Ethernet a destinatarului. O soluție este să avem undeva în sistem un fișier de configurare care transformă adresele IP în adrese Ethernet. Această soluție este posibilă, desigur, dar pentru organizații cu mii de mașini, menținerea fișierelor actualizate este o acțiune consumatoare de timp și care poate genera erori.

O soluție mai bună este ca gazda 1 să trimită un pachet de difuzare în rețeaua Ethernet întrebând: „Cine este proprietarul adresei IP 192.31.65.5?”. Pachetul de difuzare va ajunge la toate mașinile din rețeaua Ethernet 192.31.65.0 și fiecare își va verifica adresa IP. Numai gazda 2 va răspunde cu adresa sa Ethernet (*E2*). În acest mod gazda 1 află că adresa IP 192.31.65.5 este pe gazda cu adresa Ethernet *E2*. Protocolul folosit pentru a pune astfel de întrebări și a primi răspunsul se numește **ARP** (**Address Resolution Protocol** - Protocolul de rezoluție a adresei). Aproape toate mașinile din Internet îl folosesc. El este definit în RFC 826.

Avantajul folosirii ARP față de fișierele de configurare îl reprezintă simplitatea. Administratorul de sistem nu trebuie să facă prea multe, decât să atribuie fiecărei mașini o adresă IP și să hotărască măștile subrețelelor. ARP-ul face restul.

În acest punct, programele IP de pe gazda 1 construiesc un cadru Ethernet adresat către *E2*, pun pachetul IP (adresat către 193.31.65.5) în câmpul informație utilă și îl lansează pe rețeaua Ethernet. Placa Ethernet a gazdei 2 detectează acest cadru, recunoaște că este un cadru pentru ea, îl ia repede și generează o întrerupere. Driverul Ethernet extrage pachetul IP din informația utilă și îl trimite programelor IP, care văd că este corect adresat și îl prelucrează.

Pentru a face ARP-ul mai eficient sunt posibile mai multe optimizări. Pentru început, la fiecare execuție a ARP, mașina păstrează rezultatul pentru cazul în care are nevoie să contacteze din nou aceeași mașină în scurt timp. Data viitoare va găsi local corespondentul adresei, evitându-se astfel necesitatea unei a doua difuzări. În multe cazuri, gazda 2 trebuie să trimită înapoi un răspuns, ceea ce o forțează să execute ARP, pentru a determina adresa Ethernet a expeditorului. Această difuzare ARP poate fi evitată obligând gazda 1 să includă în pachetul ARP corespondența dintre adresa sa IP și adresa Ethernet. Când pachetul ARP ajunge la gazda 2, perechea (192.31.65.7, E1) este memorată local de ARP pentru o folosire ulterioară. De fapt, toate mașinile din rețeaua Ethernet pot memora această relație în memoria ARP locală.

Altă optimizare este ca fiecare mașină să difuzeze corespondența sa de adrese la pornirea mașinii. Această difuzare este realizată în general printr-un pachet ARP de căutare a propriei adrese IP. Nu ar trebui să existe un răspuns, dar un efect lateral al difuzării este introducerea unei înregistrări în memoria ascunsă ARP a tuturor. Dacă totuși sosește un răspuns (neașteptat), înseamnă că două mașini au aceeași adresă IP. Noua mașină ar trebui să-l informeze pe administratorul de sistem și să nu pornească.

Pentru a permite schimbarea relației, de exemplu, când o placă Ethernet se strică și este înlocuită cu una nouă (și astfel apare o nouă adresă Ethernet), înregistrările din memoria ascunsă ARP ar trebui să expire după câteva minute.

Să privim din nou fig. 5-62, numai că de această dată gazda 1 vrea să trimită un pachet către gazda 4 (192.31.63.8). Folosirea ARP va eșua pentru că gazda 4 nu va vedea difuzarea (ruterele nu trimit mai departe difuzările de nivel Ethernet). Există două soluții. Prima: ruterul facultății de Calculatoare poate fi configurat să răspundă la cererile ARP pentru rețeaua 193.31.63.0 (și posibil și pentru alte rețele locale). În acest caz, gazda 1 va memora local perechea (193.31.63.8, E3) și va trimite tot traficul pentru gazda 4 către ruterul local. Această soluție se numește **ARP cu intermediar** (**proxy ARP**). A doua soluție este ca gazda 1 să-și dea seama imediat că destinația se află pe o rețea aflată la distanță și să trimită tot traficul către o adresă Ethernet implicită care tratează tot traficul la distanță, în acest caz *E3*. Această soluție nu necesită ca ruterul facultății de Calculatoare să știe ce rețele la distanță deservește.

În ambele cazuri, ceea ce se întâmplă este că gazda 1 împachetează pachetul IP în câmpul informație utilă dintr-un cadru Ethernet adresat către *E3*. Când ruterul facultății de Calculatoare primeș-

te cadrul Ethernet, extrage pachetul IP din câmpul informație utilă și caută adresa IP din tabelele sale de dirijare. Descoperă că pachetele pentru rețeaua 193.31.63.0 trebuie să meargă către ruterul 192.31.60.7. Dacă nu cunoaște încă adresa FDDI a lui 193.31.60.7, difuzează un pachet ARP pe inel și află că adresa din inel este *F3*. Apoi inserează pachetul în câmpul informație utilă al unui cadru FDDI adresat către *F3* și îl transmite pe inel.

Driverul FDDI al ruterului facultății de Inginerie Electrică scoate pachetul din câmpul informație utilă și îl trimite programelor IP care văd că trebuie să trimită pachetul către 192.31.63.8. Dacă această adresă IP nu este în memoria ascunsă ARP, difuzează o cerere ARP pe rețeaua Ethernet a facultății de Inginerie Electrică și află că adresa destinație este *E6*, astfel încât construiește un cadru Ethernet adresat către *E6*, pune pachetul în câmpul informație utilă și îl trimite în rețeaua Ethernet. Când cadrul Ethernet ajunge la gazda 4, pachetul este extras din cadru și trimis programelor IP pentru procesare.

Transferul între gazda 1 și o rețea la distanță peste un WAN funcționează în esență asemănător, cu excepția că de data aceasta tabelele ruterului facultății de Calculatoare îi vor indica folosirea ruterului WAN, a cărui adresă FDDI este *F2*.

RARP, BOOTP și DHCP

ARP-ul rezolvă problema aflării adresei Ethernet corespunzătoare unei adrese IP date. Câteodată trebuie rezolvată problema inversă: dându-se o adresă Ethernet, care este adresa IP corespunzătoare? În particular, această problemă apare când se pornește o stație de lucru fără disc. O astfel de mașină va primi, în mod normal, imaginea binară a sistemului său de operare de la un server de fișiere la distanță. Dar cum își află adresa IP?

Prima soluție proiectată a fost **RARP** (**Reverse Address Resolution Protocol** - Protocol de rezoluție inversă a adresei) (definit în RFC 903). Acest protocol permite unei stații de lucru de-abia pornită să difuzeze adresa sa Ethernet și să spună: „Adresa mea Ethernet de 48 de biți este 14.04.05.18.01.25. Știe cineva adresa mea IP?" Serverul RARP vede această cerere, caută adresa Ethernet în fișierele sale de configurare și trimite înapoi adresa IP corespunzătoare.

Folosirea RARP este mai bună decât introducerea unei adrese IP în imaginea de memorie, pentru că permite ca aceeași imagine să fie folosită pe toate mașinile. Dacă adresa IP ar fi fixată înăuntrul imaginii, atunci fiecare stație de lucru ar necesita imaginea sa proprie.

Un dezavantaj al RARP este că, pentru a ajunge la serverul RARP, folosește o adresă destinație numai din 1-uri (difuzare limitată). Cu toate acestea, asemenea difuzări nu sunt propagate de rutere, așa încât este necesar un server RARP în fiecare rețea. Pentru a rezolva această problemă, a fost inventat un protocol alternativ de pornire, numit BOOTP (vezi RFC-urile 951, 1048 și 1084). Spre deosebire de RARP, acesta folosește mesaje UDP, care sunt propagate prin rutere. De asemenea furnizează unei stații de lucru fără disc informații suplimentare, care includ adresa IP a serverului de fișiere care deține imaginea de memorie, adresa IP a ruterului implicit și masca de subrețea care se folosește. BOOTP-ul este descris în RFC 951, 1048 și 1084.

O problemă serioasă cu BOOTP este că necesită configurarea manuală a corespondențelor între adresele IP și adresele Ethernet. Atunci când o nouă gazdă este adăugată la LAN, nu poate folosi BOOTP decât atunci când un administrator îi alocă o adresă IP și introduce manual (adresă Ethernet, adresă IP) în tabelele de configurare BOOTP. Pentru a elimina acest pas predispus la erori, BOOTP a fost extins și i-a fost dat un nou nume: **DHCP** (**Dynamic Host Configuration Protocol** – Protocol dinamic de configurare a gazdei). DHCP permite atât atribuirea manuală

de adrese IP, cât și atribuirea automată. Este descris în RFC-urile 2131 și 2132. În majoritatea sistemelor, a înlocuit în mare parte RARP și BOOTP.

Ca și RARP și BOOTP, DHCP este bazat pe ideea unui server special care atribuie adrese IP gazdelor care cer una. Acest server nu trebuie să se afle în același LAN cu gazda care face cererea. Deoarece serverul DHCP s-ar putea să nu fie accesibil prin difuzare, este nevoie ca în fiecare LAN să existe un **agent de legătură DHCP** (**DHCP relay agent**), așa cum se vede în fi fig. 5-63.

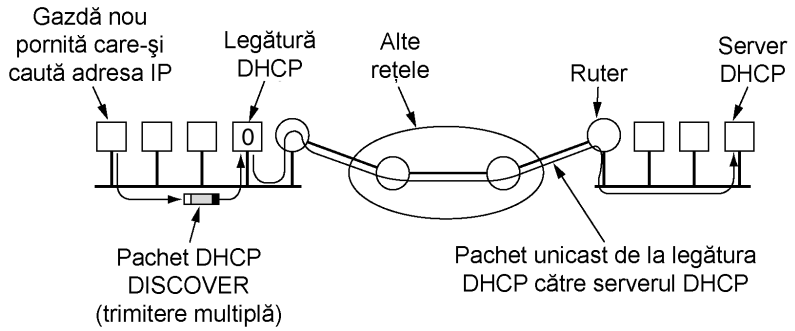

Gazdă nou pornită care-și caută adresa IP
Legătură DHCP
Alte rețele
Ruter
Server DHCP
Pachet DHCP DISCOVER (trimitere multiplă)
Pachet unicast de la legătura DHCP către serverul DHCP

Fig. 5-63. Funcționarea DHCP.

Pentru a-și afla adresa IP, o mașină tocmai pornită difuzează un pachet DHCP DISCOVER. Agentul de legătură DHCP din LAN interceptează toate difuzările DHCP. Atunci când găsește un pachet DHCP DISCOVER, îl trimite ca pachet unicast serverului DHCP, posibil într-o rețea depărtată. Singura informație de care are nevoie agentul este adresa IP a serverului DHCP.

O problemă care apare cu atribuirea automată a adreselor IP dintr-o rezervă comună este cât de mult ar trebui alocată o adresă IP . Dacă o gazdă părăsește rețeaua și nu returnează adresa sa IP serverului DHCP, acea adresă va fi pierdută permanent. După o perioadă de timp vor fi pierdute multe adrese. Pentru a preveni aceasta, atribuirea adresei IP va fi pentru o perioadă fixă de timp, o tehnică numită închiriere. Chiar înainte ca perioada de închiriere să expire, gazda trebuie să îi ceară DHCP-ului o reînnoire. Dacă nu reușește să facă cererea sau dacă cererea este respinsă, gazda nu va mai putea folosi adresa IP care îi fusese dată mai devreme.

5.5.5 Protocolul de dirijare folosit de porțile interioare: OSPF

Am terminat acum studiul protocoalelor de control ale Internetului. Este timpul să trecem la următorul subiect : rutarea în Internet. Așa cum am menționat anterior, Internet-ul este construit dintr-un număr mare de sisteme autonome. Fiecare AS este administrat de o organizație diferită și poate folosi propriul algoritm de dirijare în interior. De exemplu, rețelele interne ale companiilor *X*, *Y* și *Z* ar fi văzute ca trei AS-uri dacă toate ar fi în Internet. Toate trei pot folosi intern algoritmi de dirijare diferiți. Cu toate acestea, existența standardelor, chiar și pentru dirijarea internă, simplifică implementarea la granițele dintre AS-uri și permite reutilizarea codului. În această secțiune vom studia dirijarea în cadrul unui AS. În următoarea, vom examina dirijarea între AS-uri. Un algoritm de dirijare în interiorul unui AS este numit **protocol de porți interioare**; un algoritm de dirijare utilizat între AS-uri este numit **protocol de porți exterioare**.

Protocolul de porți interioare inițial în Internet a fost un protocol de dirijare pe baza vectorilor distanță (RIP) bazat pe algoritmul Bellman-Ford moștenit de la ARPANET. El funcționa bine în sisteme mici, dar mai puțin bine pe măsură ce AS-urile s-au extins. El suferea de asemenea de pro-

blema numărării la infinit și de o convergență slabă în general, așa că în mai 1979 a fost înlocuit de un protocol bazat pe starea legăturilor. În 1988, Internet Engineering Task Force a început lucrul la un succesor. Acel succesor, numit **OSPF** (**Open Shortest Path First** - protocol public (deschis) bazat pe calea cea mai scurtă) a devenit un standard în 1990. Mulți producători de rutere oferă suport pentru el și acesta a devenit principalul protocol de porți interioare. În cele ce urmează vom face o schiță a funcționării OSPF. Pentru informații complete, vedeți RFC 2328.

Dată fiind experiența îndelungată cu alte protocoale de dirijare, grupul care a proiectat noul protocol a avut o listă lungă de cerințe care trebuiau satisfăcute. Mai întâi, algoritmul trebuia publicat în literatura publică (open), de unde provine „O" din OSPF. O soluție în proprietatea unei companii nu era un candidat. În al doilea rând, noul protocol trebuia să suporte o varietate de metrici de distanță, incluzând distanța fizică, întârzierea ș.a.m.d. În al treilea rând, el trebuia să fie un algoritm dinamic, care să se adapteze automat și repede la schimbările în topologie.

În al patrulea rând și nou pentru OSPF, trebuia să suporte dirijarea bazată pe tipul de serviciu. Noul protocol trebuia să fie capabil să dirijeze traficul de timp real într-un mod, iar alt tip de trafic în alt mod. Protocolul IP are câmpul *Tip serviciu*, dar nici un protocol de dirijare nu-l folosea. Acest câmp a fost introdus în OSPF dar tot nu îl folosea nimeni și în cele din urmă a fost eliminat.

În al cincilea rând și în legătură cu cele de mai sus, noul protocol trebuia să facă echilibrarea încărcării, divizând încărcarea pe mai multe linii. Majoritatea protocoalelor anterioare trimit toate pachetele pe cea mai bună cale. Calea de pe locul doi nu era folosită de loc. În multe cazuri, divizarea încărcării pe mai multe linii duce la performanțe mai bune.

În al șaselea rând, era necesar suportul pentru sisteme ierarhice. Până în 1988, Internet a crescut atât de mult, încât nu se poate aștepta ca un ruter să cunoască întreaga topologie. Noul protocol de dirijare trebuia să fie proiectat astfel, încât nici un ruter să nu fie nevoit să cunoască toată topologia.

În al șaptelea rând, se cerea un minim de măsuri de securitate, pentru a evita ca studenții iubitori de distracții să păcălească ruterele trimițându-le informații de dirijare false. În fine, a fost necesară luarea de măsuri pentru a trata ruterele care au fost conectate la Internet printr-un tunel. Protocoalele precedente nu tratau bine acest caz.

OSPF suportă trei tipuri de conexiuni și rețele:

1. Linii punct-la-punct între exact două rutere.
2. Rețele multiacces cu difuzare (de exemplu, cele mai multe LAN-uri).
3. Rețele multiacces fără difuzare (de exemplu, cele mai multe WAN-uri cu comutare de pachete).

O rețea **multiacces** este o rețea care poate să conțină mai multe rutere, fiecare dintre ele putând comunica direct cu toate celelalte. Toate LAN-urile și WAN-urile au această proprietate. Fig. 5-64(a) prezintă un AS care conține toate cele trei tipuri de rețele. Observați că gazdele, în general, nu joacă un rol în OSPF.

OSPF funcționează prin abstractizarea colecției de rețele, rutere și linii reale într-un graf orientat în care fiecărui arc îi este atribuit un cost (distanță, întârziere etc.). Apoi calculează cea mai scurtă cale bazându-se pe ponderile arcelor. O conexiune serială între două rutere este reprezentată de o pereche de arce, câte unul în fiecare direcție. Ponderile lor pot fi diferite. O rețea multiacces este reprezentată de un nod pentru rețeaua însăși plus câte un nod pentru fiecare ruter. Arcele de la nodul rețea la rutere au pondere 0 și sunt omise din graf.

Fig. 5-64(b) prezintă graful pentru rețeaua din fig. 5-64(a). Ponderile sunt simetrice, dacă nu se specifică altfel. Fundamental, ceea ce face OSPF este să reprezinte rețeaua reală printr-un graf ca acesta și apoi să calculeze cea mai scurtă cale de la fiecare ruter la fiecare alt ruter.

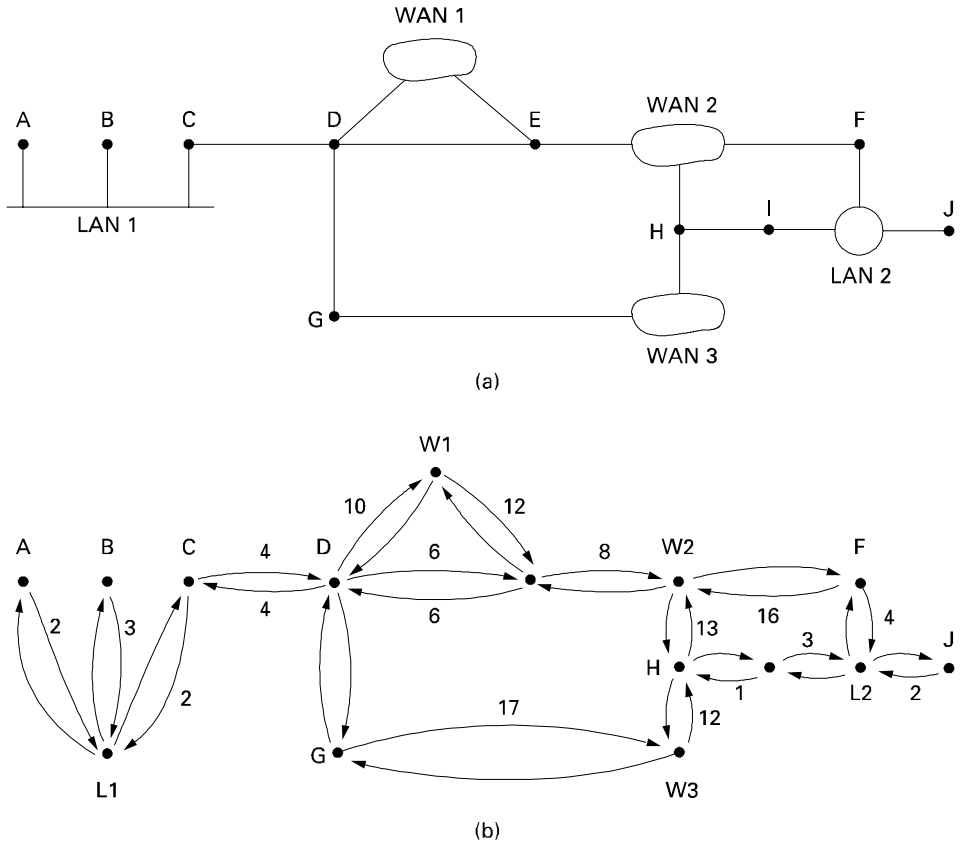


Fig. 5-64. (a) Un sistem autonom. (b) O reprezentare de tip graf a lui (a).

Multe din AS-urile din Internet sunt foarte mari și nu sunt simplu de administrat. OSPF le permite să fie divizate în **zone** numerotate, unde o zonă este o rețea sau o mulțime de rețele învecinate. Zonele nu se suprapun și nu este necesar să fie exhaustive, în sensul că unele rutere pot să nu aparțină nici unei zone. O zonă este o generalizare a unei subrețele. În afara zonei, topologia și detaliile sale nu sunt vizibile.

Orice AS are o zonă **de coloană vertebrală**, numită zona 0. Toate zonele sunt conectate la coloana vertebrală, eventual prin tunele, astfel încât este posibil să se ajungă din orice zonă din AS în orice altă zonă din AS prin intermediul coloanei vertebrale. Un tunel este reprezentat în graf ca un arc și are un cost. Fiecare ruter care este conectat la două sau mai multe zone aparține coloanei vertebrale. Analog cu celelalte zone, topologia coloanei vertebrale nu este vizibilă din afara coloanei vertebrale.

În interiorul unei zone, fiecare ruter are aceeași bază de date pentru starea legăturilor și folosește același algoritm de cea mai scurtă cale. Principala sa sarcină este să calculeze cea mai scurtă cale de la sine la fiecare alt ruter din zonă, incluzând ruterul care este conectat la coloana vertebrală, din care trebuie să existe cel puțin unul. Un ruter care conectează două zone are nevoie de bazele de date pentru ambele zone și trebuie să folosească algoritmul de cale cât mai scurtă separat pentru fiecare zonă.

În timpul operării normale pot fi necesare trei tipuri de căi: intrazonale, interzonale și interAS-uri. Rutele intrazonale sunt cele mai ușoare, din moment ce ruterul sursă știe întotdeauna calea cea

mai scurtă spre ruterul destinație. Dirijarea interzonală se desfășoară întotdeauna în trei pași: drum de la sursă la coloana vertebrală; drum de-a lungul coloanei vertebrale până la zona destinație; drum la destinație. Acest algoritm forțează o configurație de tip stea pentru OSPF, coloana vertebrală fiind concentratorul (hub), iar celelalte zone fiind spițele. Pachetele sunt dirijate de la sursă la destinație „ca atare". Ele nu sunt încapsulate sau trecute prin tunel, cu excepția cazului în care merg spre o zonă a cărei unică conexiune la coloana vertebrală este un tunel. Fig. 5-65 arată o parte a Internetului cu AS-uri și zone.

OSPF distinge patru clase de rutere:

1. Ruterele interne sunt integral în interiorul unei zone.
2. Ruterele de la granița zonei conectează două sau mai multe zone.
3. Ruterele coloanei vertebrale sunt pe coloana vertebrală.
4. Ruterele de la granița AS-urilor discută cu ruterele din alte AS-uri.

Aceste clase pot să se suprapună. De exemplu, toate ruterele de graniță fac parte în mod automat din coloana vertebrală. În plus, un ruter care este în coloana vertebrală, dar nu face parte din nici o altă zonă este de asemenea un ruter intern. Exemple din toate cele patru clase de rutere sunt ilustrate în fig. 5-65.

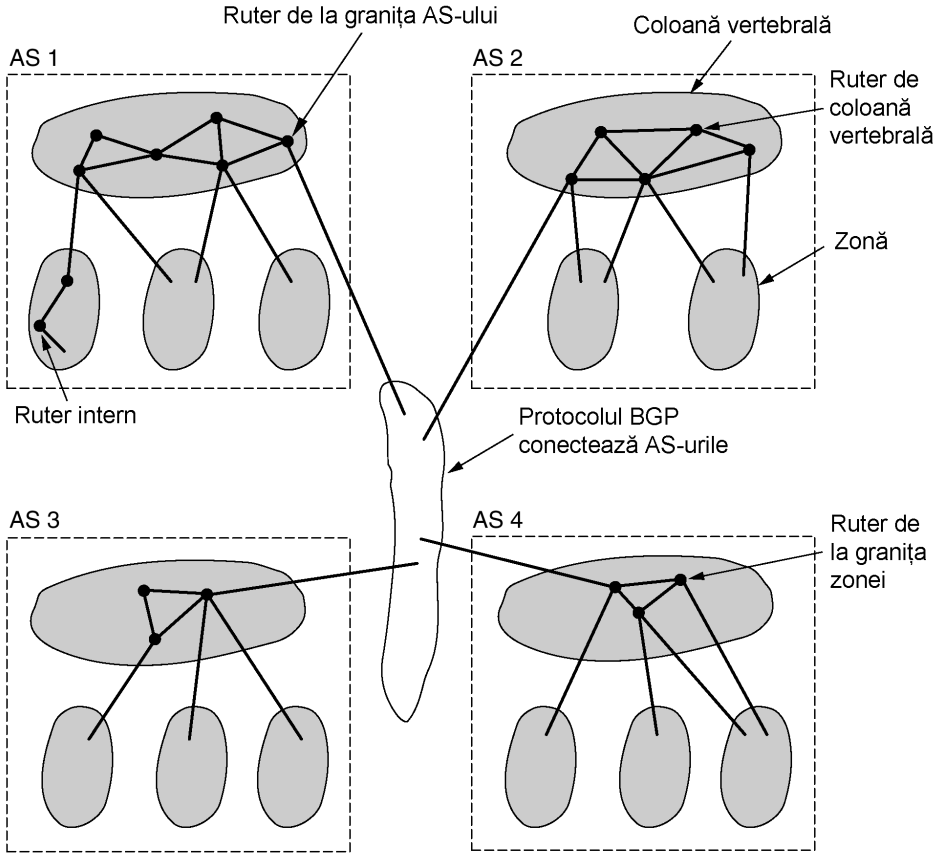


Fig. 5-65. Relația dintre AS-uri, coloane vertebrale și zone în OSPF.

Când un ruter pornește, trimite mesaje HELLO pe toate liniile sale punct-la-punct și trimite multiplu (multicast) în LAN-urile grupului compus din toate celelalte rutere. În WAN-uri, are nevoie de anumite informații de configurație, pentru a ști pe cine să contacteze. Din răspunsuri, fiecare ruter află care sunt vecinii săi. Ruterele din același LAN sunt toate vecine.

OSPF funcționează prin schimb de informații între rutere **adiacente**, care nu este același lucru cu schimbul de informații între ruterele vecine. În particular, este ineficient ca fiecare ruter dintr-un LAN să discute cu fiecare alt ruter din LAN. Pentru a evita această situație, un ruter este ales ca **ruter desemnat**. Se spune că el este adiacent cu toate celelalte rutere din LAN-ul său și schimbă informații cu ele. Ruterele vecine care nu sunt adiacente nu schimbă informații între ele. De asemenea, este actualizat în permanență și un ruter desemnat de rezervă pentru a ușura tranziția dacă ruterul desemnat primar se defectează și trebuie să fie înlocuit imediat.

În timpul funcționării normale, fiecare ruter inundă periodic cu mesaje ACTUALIZARE STARE LEGĂTURĂ (Link State Update) fiecare ruter adiacent. Acest mesaj indică starea sa și furnizează costurile folosite în baza de date topologică. Mesajele de inundare sunt confirmate pentru a le face sigure. Fiecare mesaj are un număr de secvență, astfel încât un ruter poate vedea dacă un mesaj ACTUALIZARE STARE LEGĂTURĂ este mai vechi sau mai nou decât ceea ce are deja. De asemenea, ruterele trimit aceste mesaje când o linie cade sau își revine sau când costul acesteia se modifică.

Mesajele DESCRIERE BAZA DE DATE (Database Description) dau numerele de secvență pentru toate intrările de stare a liniei deținute actual de emițător. Prin compararea valorilor proprii cu acelea ale emițătorului, receptorul poate determina cine are cea mai recentă valoare. Aceste mesaje sunt folosite când o linie este refăcută.

Fiecare partener poate cere informații de stare a legăturii de la celălalt folosind mesaje CERERE STARE LEGĂTURĂ (Link State Request). Rezultatul concret al acestui algoritm este că fiecare pereche de rutere adiacente verifică să vadă cine are cele mai recente date și astfel, noua informație este răspândită în zonă. Toate aceste mesaje sunt trimise ca simple pachete IP. Cele cinci tipuri de mesaje sunt rezumate în fig. 5-66.

Tip mesaj	Descriere
Hello	Folosit pentru descoperirea vecinilor
Actualizare Stare Legătură	Emițătorul furnizează vecinilor săi costurile sale
Confirmare Stare Legătură	Confirmă actualizarea stării legăturii
Descriere Bază de Date	Anunță ce actualizări are emițătorul
Cerere Stare Legătură	Cere informații de la partener

Fig. 5-66. Cele cinci tipuri de mesaje OSPF.

În final, putem să asamblăm toate piesele. Folosind inundarea, fiecare ruter informează toate celelalte rutere din zona sa despre vecinii și costurile sale. Această informație permite fiecărui ruter să construiască graful zonei (zonelor) sale și să calculeze cea mai scurtă cale. Zona de coloană vertebrală face și ea același lucru. În plus, ruterele de coloană vertebrală acceptă informația de la ruterele zonei de graniță cu scopul de a calcula cea mai bună cale de la fiecare ruter de coloană vertebrală către fiecare alt ruter. Această informație este propagată înapoi către ruterele zonei de graniță, care o fac publică în zonele lor. Folosind această informație, un ruter gata să trimită un pachet interzonal poate selecta cel mai bun ruter de ieșire către coloana vertebrală.

5.6.5 Protocolul de dirijare pentru porți externe: BGP

În cadrul unui singur AS, protocolul de dirijare recomandat este OSPF (deși, desigur, nu este singurul folosit). Între AS-uri se folosește un protocol diferit, **BGP** (**Border Gateway Protocol** - Protocolul porților de graniță). Între AS-uri este necesar un protocol diferit pentru că scopurile unui protocol pentru porți interioare și ale unui protocol pentru porți exterioare sunt diferite. Tot ce trebuie să facă un protocol pentru porți interioare este să mute pachetele cât mai eficient de la sursă la destinație. El nu trebuie să-și facă probleme cu politica.

Ruterele ce folosesc protocolul de porți exterioare trebuie să țină cont într-o mare măsură de politică (Metz, 2001). De exemplu, un AS al unei corporații poate dori facilitatea de a trimite pachete oricărui sit Internet și să recepționeze pachete de la orice sit Internet. Cu toate acestea, poate să nu dorească să asigure tranzitarea pentru pachetele originare dintr-un AS străin destinate unui AS străin diferit, chiar dacă prin AS-ul propriu trece cea mai scurtă cale dintre cele două AS-uri străine („Asta este problema lor, nu a noastră."). Pe de altă parte, poate fi dispus să asigure tranzitarea pentru vecinii săi, sau chiar pentru anumite AS-uri care au plătit pentru acest serviciu. Companiile telefonice, de exemplu, pot fi fericite să acționeze ca un purtător pentru clienții lor, dar nu și pentru alții. Protocoalele pentru porți externe, în general și BGP în particular, au fost proiectate pentru a permite forțarea multor tipuri de politici de dirijare pentru traficul între AS-uri.

Politicile tipice implică considerații politice, de securitate sau economice. Câteva exemple de constrângeri de dirijare sunt:

1. Nu se tranzitează traficul prin anumite AS-uri.
2. Nu se plasează Irak-ul pe o rută care pornește din Pentagon.
3. Nu se folosesc Statele Unite pentru a ajunge din Columbia Britanică în Ontario.
4. Albania este tranzitată numai dacă nu există altă alternativă către destinație.
5. Traficul care pleacă sau ajunge la IBM nu trebuie să tranziteze Microsoft.

În mod obișnuit politicile sunt configurate manual în fiecare ruter BGP (sau sunt incluse folosind un anumit tip de script). Ele nu sunt parte a protocolului însuși.

Din punctul de vedere al unui ruter BGP, lumea constă din AS-uri și liniile care le conectează. Două AS-uri sunt considerate conectate dacă există o linie între două rutere de graniță din fiecare. Dat fiind interesul special al BGP-ului pentru traficul în tranzit, rețelele sunt grupate în trei categorii. Prima categorie este cea a **rețelelor ciot** (**stub networks**), care au doar o conexiune la graful BGP. Acestea nu pot fi folosite pentru traficul în tranzit pentru că nu este nimeni la capătul celălalt. Apoi vin **rețelele multiconectate**. Acestea pot fi folosite pentru traficul în tranzit, cu excepția a ceea ce ele refuză. În final, sunt **rețele de tranzit**, cum ar fi coloanele vertebrale, care sunt doritoare să manevreze pachetele altora, eventual cu unele restricții, și de obicei pentru o plată.

Perechile de rutere BGP comunică între ele stabilind conexiuni TCP. Operarea în acest mod oferă comunicație sigură și ascunde toate detaliile rețelelor traversate.

BGP este la bază un protocol bazat pe vectori distanță, dar destul de diferit de majoritatea celorlalte cum ar fi RIP. În loc să mențină doar costul până la fiecare destinație, fiecare ruter BGP memorează calea exactă folosită. Similar, în loc să trimită periodic fiecărui vecin costul său estimat către fiecare destinație posibilă, fiecare ruter BGP comunică vecinilor calea exactă pe care o folosește.

Ca exemplu, să considerăm ruterele BGP din fig. 5-67(a). În particular, să considerăm tabela de dirijare a lui *F*. Să presupunem că el folosește calea *FGCD* pentru a ajunge la *D*. Când vecinii îi dau

informațiile de dirijare, ei oferă căile lor complete, ca în fig. 5-67(b) (pentru simplitate, este arătată doar destinația *D*).

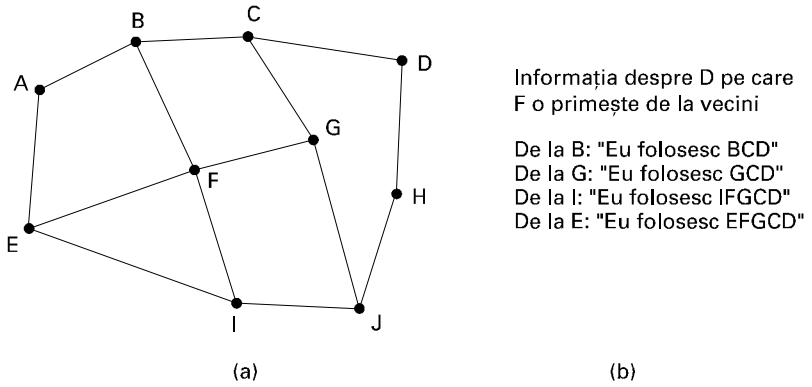


Fig. 5-67. (a) O mulțime de rutere BGP. (b) Informația trimisă lui *F*.

După ce sosesc toate căile de la vecini, *F* le examinează pentru a vedea care este cea mai bună. El elimină imediat căile de la *I* și *E*, din moment ce aceste căi trec chiar prin F. Alegerea este apoi între B și G. Fiecare ruter BGP conține un modul care examinează căile către o destinație dată și le atribuie scoruri, întorcând, pentru fiecare cale, o valoare pentru „distanța" către acea destinație. Orice cale care violează o constrângere politică primește automat un scor infinit. Apoi, ruterul adoptă calea cu cea mai scurtă distanță. Funcția de acordare a scorurilor nu este parte a protocolului BGP și poate fi orice funcție doresc administratorii de sistem.

BGP rezolvă ușor problema numără-la-infinit care chinuie alți algoritmi bazați pe vectori distanță. De exemplu, să presupunem că *G* se defectează sau că linia *FG* cade. Atunci *F* primește căi de la ceilalți trei vecini rămași. Aceste rute sunt *BCD*, *IFGCD* și *EFGCD*. El poate observa imediat că ultimele două căi sunt inutile, din moment ce trec chiar prin *F*, așa că alege *FBCD* ca noua sa cale. Alți algoritmi bazați pe vectori distanță fac de multe ori alegerea greșită, pentru că ei nu pot spune care din vecinii lor au căi independente către destinație și care nu. Definirea BGP-ului se găsește în RFC 1771 și 1774.

5.6.6 Trimiterea multiplă în Internet

Comunicația IP normală este între un emițător și un receptor. Cu toate acestea, pentru unele aplicații, este util ca un proces să fie capabil să trimită simultan unui număr mare de receptori. Exemple sunt actualizarea bazelor de date distribuite multiplicate, transmiterea cotărilor de la bursă mai multor agenți de bursă, tratarea convorbirilor telefonice de tipul conferințelor digitale (așadar cu mai mulți participanți).

IP-ul suportă trimiterea multiplă (multicast), folosind adrese de clasă D. Fiecare adresă de clasă D identifică un grup de gazde. Pentru identificarea grupurilor sunt disponibili douăzeci și opt de biți, așa încât pot exista în același moment peste 250 milioane de grupuri. Când un proces trimite un pachet unei adrese de clasă D, se face cea mai bună încercare pentru a-l livra tuturor membrilor grupului adresat, dar nu sunt date garanții. Unii membri pot să nu primească pachetul.

Sunt suportate două tipuri de adrese de grup: adrese permanente și adrese temporare. Un grup permanent există întotdeauna și nu trebuie configurat. Fiecare grup permanent are o adresă de grup permanentă. Câteva exemple de adrese de grup permanente sunt:

224.0.0.1 Toate sistemele dintr-un LAN.
224.0.0.2 Toate ruterele dintr-un LAN.
224.0.0.5 Toate ruterele OSPF dintr-un LAN.
224.0.0.6 Toate ruterele desemnate dintr-un LAN.

Grupurile temporare trebuie create înainte de a fi utilizate. Un proces poate cere gazdei sale să intre într-un anume grup. De asemenea, el poate cere gazdei sale să părăsească grupul. Când ultimul proces părăsește un grup, acel grup nu mai este prezent pe calculatorul său gazdă. Fiecare gazdă memorează căror grupuri aparțin la un moment dat procesele sale.

Trimiterea multiplă este implementată de rutere speciale de trimitere multiplă, care pot sau nu coabita cu ruterele standard. Cam o dată pe minut, fiecare ruter de trimitere multiplă face o trimitere multiplă hardware (la nivel legătură de date) pentru gazdele din LAN-ul său (adresa 224.0.0.1), cerându-le să raporteze căror grupuri aparțin procesele lor la momentul respectiv. Fiecare gazdă trimite înapoi răspunsuri pentru toate adresele de clasă D de care este interesată.

Aceste pachete de întrebare și răspuns folosesc un protocol numit **IGMP** (**Internet Group Management Protocol** - Protocol de gestiune a grupurilor Internet), care se aseamănă întrucâtva cu ICMP. El are numai două tipuri de pachete: întrebare și răspuns, fiecare cu un format fix, simplu, care conține unele informații de control în primul cuvânt al câmpului informație utilă și o adresă de clasă D în al doilea cuvânt. El este descris în RFC 1112.

Dirijarea cu trimitere multiplă este realizată folosind arbori de acoperire. Fiecare ruter de dirijare multiplă schimbă informații cu vecinii săi, folosind un protocol modificat bazat pe vectori distanță cu scopul ca fiecare să construiască pentru fiecare grup un arbore de acoperire care să acopere toți membrii grupului. Pentru a elimina ruterele și rețelele neinteresate de anumite grupuri, se utilizează diverse optimizări de reducere a arborelui. Pentru evitarea deranjării nodurilor care nu fac parte din arborele de acoperire, protocolul folosește intensiv trecerea prin tunel.

5.6.7 IP mobil

Mulți utilizatori ai Internet-ului au calculatoare portabile și vor să rămână conectați la Internet atunci când vizitează un sit Internet aflat la distanță, și chiar și pe drumul dintre cele două. Din nefericire, sistemul de adresare IP face lucrul la depărtare de casă mai ușor de zis decât de făcut. În această secțiune vom examina problema și soluția. O descriere mai detaliată este dată în (Perkins, 1998a).

Problema apare chiar în schema de adresare. Fiecare adresă IP conține un număr de rețea și un număr de gazdă. De exemplu, să considerăm mașina cu adresa IP 160.80.40.20/16. Partea 160.80 indică numărul de rețea (8272 în notație zecimală). Ruterele din toată lumea au tabele de dirijare care spun ce linie se folosește pentru a ajunge la rețeaua 160.80. Oricând vine un pachet cu adresa IP destinație de forma 160.80.xxx.yyy, pachetul pleacă pe respectiva linie.

Dacă, dintr-o dată, mașina cu adresa respectivă este transferată într-un alt loc din Internet, pachetele vor continua să fie dirijate către LAN-ul (sau ruterul) de acasă. Proprietarul nu va mai primi poștă electronică și așa mai departe. Acordarea unei noi adrese IP mașinii, adresă care să corespundă cu noua sa locație, nu este atractivă pentru că ar trebui să fie informate despre schimbare un mare număr de persoane, programe și baze de date.

O altă abordare este ca ruterele să facă dirijarea folosind adresa IP completă, în loc să folosească numai adresa rețelei. Cu toate acestea, această strategie ar necesita ca fiecare ruter să aibă milioane de intrări în tabele, la un cost astronomic pentru Internet.

Când oamenii au început să ceară posibilitatea de a-și conecta calculatoarele portabile oriunde s-ar duce, IETF a constituit un Grup de Lucru pentru a găsi o soluție. Grupul de Lucru a formulat rapid un număr de obiective considerate necesare în orice soluție. Cele majore au fost:

1. Fiecare gazdă mobilă trebuie să fie capabilă sa folosească adresa sa IP de baza oriunde.
2. Nu au fost permise schimbări de programe pentru gazdele fixe.
3. Nu au fost permise schimbări pentru programele sau tabelele ruterelor.
4. Cele mai multe pachete pentru gazdele mobile nu ar trebui să facă ocoluri pe drum.
5. Nu trebuie să apară nici o suprasolicitare când o gazdă mobilă este acasă.

Soluția aleasă este cea descrisă în secțiunea 5.2.8. Pentru a o recapitula pe scurt, fiecare sit care dorește să permită utilizatorilor săi să se deplaseze trebuie să asigure un agent local. Fiecare sit care dorește să permită accesul vizitatorilor trebuie să creeze un agent pentru străini. Când o gazdă mobilă apare într-un sit străin, ea contactează gazda străină de acolo și se înregistrează. Gazda străină contactează apoi agentul local al utilizatorului și îi dă o **adresă a intermediarului**, în mod normal adresa IP proprie a agentului pentru străini.

Când un pachet ajunge în LAN-ul de domiciliu al utilizatorului, el vine la un ruter atașat la LAN. Apoi ruterul încearcă să localizeze gazda în mod uzual, prin difuzarea unui pachet ARP întrebând, de exemplu: „Care este adresa Ethernet a lui 160.80.40.20?” Agentul local răspunde la această întrebare dând propria adresă Ethernet. Apoi ruterul trimite pachetele pentru 160.80.40.20 la agentul local. El, în schimb, le trimite prin tunel la adresa intermediarului prin încapsularea lor în câmpul informație utilă al unui pachet IP adresat agentului pentru străini. După aceasta, agentul pentru străini le desface și le livrează la adresa de nivel legătură de date a gazdei mobile. În plus, agentul local dă emițătorului adresa intermediarului, așa încât viitoarele pachete pot fi trimise prin tunel direct la agentul pentru străini. Această soluție răspunde tuturor cerințelor expuse mai sus.

Probabil că merită menționat un mic amănunt. În momentul în care gazda mobilă se mută, probabil că ruterul are adresa ei Ethernet (care în curând va fi invalidă) memorată în memoria ascunsă. Pentru a înlocui această adresă Ethernet cu cea a agentului local, se folosește un truc numit **ARP gratuit**. Acesta este un mesaj special, nesolicitat, către ruter pe care îl determină să schimbe o anumită intrare din memoria ascunsă, în acest caz cea a gazdei mobile care urmează să plece. Când, mai târziu, gazda mobilă se întoarce, este folosit același truc pentru a actualiza din nou memoria ascunsă a ruterului.

Nu există nimic în proiect care să împiedice o gazdă mobilă să fie propriul său agent pentru străini, dar această abordare funcționează numai dacă gazda mobilă (în postura sa de agent pentru străini) este conectată logic în Internet la situl său curent. De asemenea, ea trebuie să fie capabilă să obțină pentru folosire o adresă (temporară) de intermediar. Acea adresă IP trebuie să aparțină LAN-ului în care este atașată în mod curent.

Soluția IETF pentru gazde mobile rezolvă un număr de alte probleme care nu au fost menționate până acum. De exemplu, cum sunt localizați agenții? Soluția este ca fiecare agent să-și difuzeze periodic adresa și tipul de serviciu pe care dorește să-l ofere (de exemplu, agent local, pentru străini sau amândouă). Când o gazdă mobilă ajunge undeva, ea poate asculta așteptând aceste difuzări, numite **anunțuri**. Ca o alternativă, ea poate difuza un pachet prin care își anunță sosirea și să spere că agentul pentru străini local îi va răspunde.

O altă problemă care a trebuit rezolvată este cum să se trateze gazdele mobile nepoliticoase, care pleacă fără să spună la revedere. Soluția este ca înregistrarea să fie valabilă doar pentru un interval de timp fixat. Dacă nu este reîmprospătată periodic, ea expiră și ca urmare gazda străină poate să-și curețe tabelele.

O altă problemă este securitatea. Când un agent local primește un mesaj care-i cere să fie amabil să retrimită toate pachetele Robertei la o anume adresă IP, ar fi mai bine să nu se supună decât dacă este convins că Roberta este sursa acestei cereri și nu altcineva încercând să se dea drept ea. În acest scop sunt folosite protocoale criptografice de autentificare. Vom studia asemenea protocoale în cap. 8.

Un punct final adresat de Grupul de Lucru se referă la nivelurile de mobilitate. Să ne imaginăm un avion cu o rețea Ethernet la bord folosită de către calculatoarele de navigare și de bord. În această rețea Ethernet există un ruter standard, care discută cu Internet-ul cablat de la sol printr-o legătură radio. Într-o bună zi, unui director de marketing isteț îi vine ideea să instaleze conectoare Ethernet în toate brațele fotoliilor, astfel încât și pasagerii cu gazde mobile să se poată conecta.

Acum avem două niveluri de mobilitate: calculatoarele proprii ale aeronavei, care sunt staționare în raport cu rețeaua Ethernet și calculatoarele pasagerilor, care sunt mobile în raport cu ea. În plus, ruterul de la bord este mobil în raport cu ruterele de la sol. Mobilitatea în raport cu un sistem care este la rândul său mobil este tratată folosind recursiv tunele.

5.6.8 IPv6

În timp ce CIDR și NAT îi mai pot acorda câțiva ani, toată lumea își dă seama că zilele IP-ului în forma curentă (IPv4) sunt numărate. În plus față de aceste probleme tehnice, există un alt aspect întrezărit în fundal. La începuturile sale, Internet-ul a fost folosit în mare măsură de universități, industria de vârf și de guvernul Statelor Unite (în mod special de Departamentul Apărării). O dată cu explozia interesului față de Internet începând de la mijlocul anilor 1990, a început să fie utilizat de un grup diferit de persoane, în special persoane cu cerințe diferite. Pe de o parte, numeroase persoane cu calculatoare portabile fără fir îl folosesc pentru a ține legătura cu baza de acasă. Pe de altă parte, o dată cu iminenta convergență a industriilor calculatoarelor, comunicațiilor și a distracțiilor, s-ar putea să nu mai fie mult până când fiecare telefon sau televizor din lume va fi un nod Internet, producând un miliard de mașini folosite pentru audio și video la cerere. În aceste condiții, a devenit clar că IP-ul trebuie să evolueze și să devină mai flexibil.

Observând aceste probleme la orizont, IETF a început să lucreze în 1990 la o nouă versiune de IP, una care să nu își epuizeze niciodată adresele, să rezolve o gamă largă de alte probleme și să fie totodată mai flexibilă și mai eficientă. Obiectivele majore au fost:

1. Să suporte miliarde de gazde, chiar cu alocare ineficientă a spațiului de adrese.
2. Să reducă dimensiunea tabelelor de dirijare.
3. Să simplifice protocolul, pentru a permite ruterelor să proceseze pachetele mai rapid.
4. Să asigure o securitate mai bună (autentificare și confidențialitate) față de IP-ul curent.
5. Să acorde o mai mari atenție tipului de serviciu, în special pentru datele de timp real.
6. Să ajute trimiterea multiplă, permițând specificarea de domenii.
7. Să creeze condițiile pentru ca o gazdă să poată migra fără schimbarea adresei sale.
8. Să permită evoluția protocolului în viitor.
9. Să permită coexistența noului și vechiului protocol pentru câțiva ani.

Pentru a găsi un protocol care să îndeplinească toate aceste cerințe, IETF a emis o cerere de propuneri și discuții în RFC 1550. Au fost primite douăzeci și unu de răspunsuri, nu toate din ele propuneri complete. Până în decembrie 1992, au ajuns pe masa discuțiilor șapte propuneri serioase. Ele variau de la efectuarea de mici cârpeli la IP până la renunțarea completă la el și înlocuirea cu un protocol complet nou.

O propunere a fost folosirea TCP peste CLNP, care cu cei 160 de biți de adresă ai săi ar fi oferit spațiu de adrese suficient pentru totdeauna și ar fi unificat două protocoale majore de nivel rețea. Cu toate acestea, multe persoane au simțit că aceasta ar fi fost o acceptare a faptului că, de fapt, a fost făcut ceva chiar bine în lumea OSI, o afirmație considerată incorectă din punct de vedere politic în cercurile Internet. CLNP a fost modelat apropiat de IP, așa încât cele două nu sunt chiar atât de diferite. De fapt, protocolul care a fost ales în final diferă de IP cu mult mai mult decât diferă CLNP. O altă lovitură împotriva CLNP a fost slabul suport pentru tipuri de servicii, ceva necesar pentru transmiterea eficientă de multimedia.

Trei din cele mai bune propuneri au fost publicate în *IEEE Network* (Deering, 1993; Francis, 1993 și Katz și Ford, 1993). După multe discuții, revizii și manevre de culise, a fost selectată o versiune combinată modificată a propunerilor lui Deering și Francis, până atunci numită **SIPP** (**Simple Internet Protocol Plus** - Protocol simplu, îmbunătățit, pentru Internet) și i s-a dat numele de **IPv6**.

IPv6 îndeplinește obiectivele destul de bine. El menține caracteristicile bune ale IP-ului, le elimină sau atenuează pe cele rele și adaugă unele noi acolo unde este nevoie. În general, IPv6 nu este compatibil cu IPv4, dar el este compatibil cu celelalte protocoale Internet auxiliare, incluzând TCP, UDP, ICMP, IGMP, OSPF, BGP și DNS, câteodată fiind necesare mici modificări (majoritatea pentru a putea lucra cu adrese mai lungi). Principalele trăsături ale IPv6 sunt discutate mai jos. Mai multe informații despre el pot fi găsite în RFC 2460 până la RFC 2466.

În primul rând și cel mai important, IPv6 are adrese mai lungi decât IPv4. Ele au o lungime de 16 octeți, ceea ce rezolvă problema pentru a cărei soluționare a fost creat IPv6: să furnizeze o sursă efectiv nelimitată de adrese Internet. În curând vom spune mai multe despre adrese.

A doua mare îmbunătățire a lui IPv6 este simplificarea antetului. El conține numai 7 câmpuri (față de 13 în IPv4). Această schimbare permite ruterelor să prelucreze pachetele mai rapid, îmbunătățind astfel productivitatea și întârzierea. De asemenea, vom discuta în curând și antetul.

A treia mare îmbunătățire a fost suportul mai bun pentru opțiuni. Această schimbare a fost esențială în noul antet, deoarece câmpurile care erau necesare anterior sunt acum opționale. În plus, modul în care sunt reprezentate opțiunile este diferit, ușurând ruterelor saltul peste opțiunile care nu le sunt destinate. Această caracteristică accelerează timpul de prelucrare a pachetelor.

Un al patrulea domeniu în care IPv6 reprezintă un mare progres este în securitate. IETF a avut porția sa de povești de ziar despre copii precoce de 12 ani care își folosesc calculatoarele personale pentru a sparge bănci sau baze militare în tot Internet-ul. A existat un sentiment puternic că ar trebui făcut ceva pentru a îmbunătăți securitatea. Autentificarea și confidențialitatea sunt trăsături cheie ale noului IP. Ulterior ele au fost adaptate și în IPv4, astfel că în domeniul securității diferențele nu mai sunt așa de mari.

În final, a fost acordată o mai mare atenție calității serviciilor. În trecut s-au făcut eforturi, fără prea mare tragere de inimă, dar acum, o dată cu creșterea utilizării multimedia în Internet, presiunea este și mai mare.

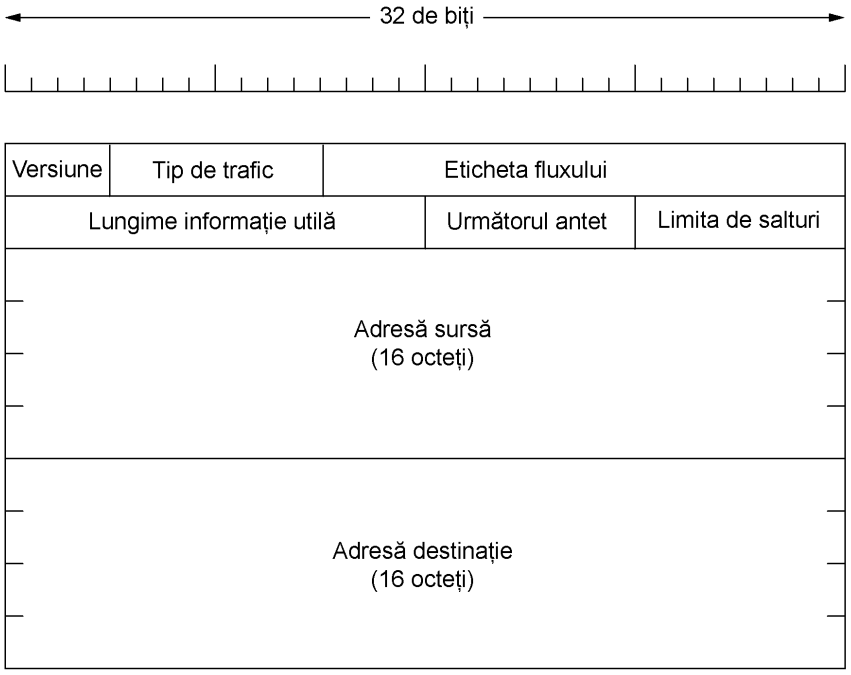


Fig. 5-68. Antetul fix IPv6 (obligatoriu).

Antetul principal IPv6

Antetul IPv6 este prezentat în fig. 5-68. Câmpul *Versiune* este întotdeauna 6 pentru IPv6 (și 4 pentru IPv4). În timpul perioadei de tranziție de la IPv4, care va lua probabil un deceniu, ruterele vor fi capabile să examineze acest câmp pentru a spune ce tip de pachet analizează. Ca un efect lateral, acest test irosește câteva instrucțiuni pe drumul critic, așa încât multe implementări vor încerca să-l evite prin folosirea unui câmp din antetul legăturii de date ca să diferențieze pachetele IPv4 de pachetele IPv6. În acest mod, pachetele pot fi transmise direct rutinei de tratare de nivel rețea corecte. Cu toate acestea, necesitatea ca nivelul legătură de date să cunoască tipurile pachetelor nivelului rețea contravine complet principiul de proiectare care spune că fiecare nivel nu trebuie să cunoască semnificația biților care îi sunt dați de către nivelul de deasupra. Discuțiile dintre taberele „Fă-o corect" și „Fă-o repede" vor fi, fără îndoială, lungi și virulente.

Câmpul *Tip de trafic* (*Traffic class*) este folosit pentru a distinge între pachetele care au diverse cerințe de livrare în timp real. Un câmp cu acest scop a existat în IP de la început, dar a fost implementat sporadic de către rutere. În acest moment se desfășoară experimente pentru a determina cum poate fi utilizat cel mai bine pentru transmisii multimedia.

Câmpul *Eticheta fluxului* este încă experimental, dar va fi folosit pentru a permite unei surse și unei destinații să stabilească o pseudo-conexiune cu proprietăți și cerințe particulare. De exemplu, un șir de pachete de la un proces de pe o anumită gazdă sursă către un anumit proces pe o anumită gazdă destinație poate avea cerințe de întârziere stricte și din acest motiv necesită capacitate de transmisie rezervată. Fluxul poate fi stabilit în avans și poate primi un identificator. Când apare un pachet cu o *Etichetă a fluxului* diferită de zero, toate ruterele pot să o caute în tabelele interne pentru

a vedea ce tip de tratament special necesită. Ca efect, fluxurile sunt o încercare de a combina două moduri: flexibilitatea unei subrețele cu datagrame și garanțiile unei subrețele cu circuite virtuale.

Fiecare flux este desemnat de adresa sursă, adresa destinație și numărul de flux, așa încât, între o pereche dată de adrese IP pot exista mai multe fluxuri active în același timp . De asemenea, în acest mod, chiar dacă două fluxuri venind de la gazde diferite, dar cu același număr de flux trec prin același ruter, ruterul va fi capabil să le separe folosind adresele sursă și destinație. Se așteaptă ca numerele de flux să fie alese aleator, în loc de a fi atribuite secvențial începând cu 1, pentru că se așteaptă ca ruterele să le folosească în tabele de dispersie.

Câmpul *Lungimea informației utile* spune câți octeți urmează după antetul de 40 de octeți din fig. 5-68. Numele a fost schimbat față de câmpul *Lungime totală* din IPv4 deoarece semnificația este ușor modificată: cei 40 de octeți nu mai sunt parte a lungimii (așa cum erau înainte).

Câmpul *Antetul următor* dă de gol proiectanții. Motivul pentru care antetul a putut fi simplificat este că există antete de extensie suplimentare (opționale). Acest câmp spune care din cele șase antete (actuale) de extensie, dacă există vreunul, urmează după cel curent. Dacă acest antet este ultimul antet IP, câmpul *Antetul următor* spune cărui tip de protocol (de exemplu TCP, UDP) i se va transmite pachetul.

Câmpul *Limita salturilor* este folosit pentru a împiedica pachetele să trăiască veșnic. El este, în practică, identic cu câmpul *Timp de viață* din IPv4, și anume un câmp care este decrementat la fiecare salt dintr-o rețea în alta. În teorie, în IPv4 era un timp în secunde, dar nici un ruter nu-l folosea în acest mod, așa încât numele a fost modificat pentru a reflecta modul în care este de fapt folosit.

Apoi urmează câmpurile *Adresă sursă* și *Adresă destinație*. Propunerea originală a lui Deering, SIP, folosea adrese de 8 octeți, dar în timpul procesului de evaluare, multe persoane au simțit că adresele de 8 octeți s-ar putea epuiza în câteva decenii, în timp ce adresele de 16 octeți nu s-ar epuiza niciodată. Alte persoane au argumentat că 16 octeți ar fi un exces, în timp ce alții încurajau folosirea adreselor de 20 de octeți pentru a fi compatibile cu protocolul datagramă OSI. O altă grupare dorea adrese de dimensiune variabilă. După multe discuții, s-a decis că adresele cu lungime fixă de 16 octeți sunt cel mai bun compromis.

Pentru scrierea adreselor de 16 octeți a fost inventată o nouă notație. Ele sunt scrise ca opt grupuri de câte patru cifre hexazecimale cu semnul : (două puncte) între grupuri, astfel:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Din moment ce multe adrese vor avea multe zerouri în interiorul lor, au fost autorizate trei optimizări. Mai întâi, zerourile de la începutul unui grup pot fi omise, astfel încât 0123 poate fi scris ca 123. În al doilea rând, unul sau mai multe grupuri de 16 zerouri pot fi înlocuite de o pereche de semne două puncte (:). Astfel, adresa de mai sus devine acum

8000::123:4567:89AB:CDEF

În final, adresele IPv4 pot fi scrise ca o pereche de semne două puncte și un număr zecimal în vechea formă cu punct, de exemplu

::192.31.20.46

Probabil că nu este necesar să fim atât de expliciți asupra acestui lucru, dar există o mulțime de adrese de 16 octeți. Mai exact, sunt 2^{128} adrese, care reprezintă aproximativ $3x10^{38}$. Dacă întreaga planetă, pământ și apă, ar fi acoperite cu calculatoare, IPv6 ar permite $7x10^{23}$ adrese IP pe metru pătrat. Studenții de la chimie vor observa că acest număr este mai mare decât numărul lui Avogadro. Deși nu a

existat intenția de a da fiecărei molecule de pe suprafața planetei adresa ei IP, nu suntem chiar așa de departe de aceasta.

În practică, spațiul de adrese nu va fi folosit eficient, așa cum nu este folosit spațiul de adrese al numerelor de telefon (prefixul pentru Manhattan, 212, este aproape plin, dar cel pentru Wyoming, 307, este aproape gol). În RFC 3194, Durand și Huitema a calculat că, folosind ca referință alocarea numerelor de telefon, chiar și în cel mai pesimist scenariu, vor fi totuși mult peste 1000 de adrese IP pe metru pătrat de suprafață planetară (pământ sau apă). În orice scenariu credibil, vor fi trilioane de adrese pe metru pătrat. Pe scurt, pare improbabil că vom epuiza adresele în viitorul previzibil.

Este instructiv să comparăm antetul IPv4 (fig. 5-53) cu antetul IPv6 (fig. 5-68) pentru a vedea ce a fost eliminat în IPv6. Câmpul *IHL* a dispărut pentru că antetul IPv6 are o lungime fixă. Câmpul *Protocol* a fost scos pentru că în câmpul *Antetul următor* se indică ce urmează după ultimul antet IP (de exemplu, un segment TCP sau UDP).

Toate câmpurile referitoare la fragmentare au fost eliminate, deoarece IPv6 are o abordare diferită a fragmentării. Pentru început, toate gazdele și ruterele care sunt conforme cu IPv6 trebuie să determine dinamic mărimea datagramei care va fi folosită. Această regulă face ca, de la început, fragmentarea să fie mai puțin probabilă. De asemenea minimul a fost mărit de la 576 la 1280 pentru a permite date de 1024 de octeți și mai multe antete. În plus, când o gazdă trimite un pachet IPv6 care este prea mare, ruterul care este incapabil să îl retransmită trimite înapoi un mesaj de eroare în loc să fragmenteze pachetul. Acest mesaj de eroare îi spune gazdei să spargă toate pachetele viitoare către acea destinație. Este mult mai eficient să obligi gazdele să trimită de la bun început pachete corecte dimensional, decât să obligi ruterele să le fragmenteze din mers.

În sfârșit, câmpul *Sumă de control* este eliminat deoarece calculul acesteia reduce mult performanțele. Datorită rețelelor fiabile folosite acum, combinate cu faptul că nivelurile de legătură de date și de transport au în mod normal propriile sume de control, valoarea a încă unei sume de control nu merita prețul de performanță cerut. Eliminarea tuturor acestor caracteristici a avut ca rezultat un protocol de nivel rețea simplu și eficient. Astfel, obiectivul lui IPv6 – un protocol rapid, dar flexibil, cu o bogăție de spațiu de adrese – a fost atins prin acest proiect.

Antete de extensie

Câteva din câmpurile care lipsesc sunt încă necesare ocazional, astfel încât IPv6 a introdus conceptul de **antet de extensie** (opțional). Aceste antete pot fi furnizate pentru a oferi informații suplimentare, dar codificate într-un mod eficient. În prezent sunt definite șase tipuri de antete de extensie, prezentate în fig. 5-69. Fiecare este opțional, dar dacă sunt prezente mai multe, ele trebuie să apară imediat după antetul fix și, preferabil, în ordinea prezentată.

Antet de extensie	Descriere
Opțiuni salt-după-salt	Diverse informații pentru rutere
Opțiuni pentru destinație	Informații suplimentare pentru destinație
Dirijare	Calea, parțială sau totală, de urmat
Fragmentare	Gestiunea fragmentelor datagramelor
Autentificare	Verificarea identității emițătorului
Informație de siguranță criptată	Informații despre conținutul criptat

Fig. 5-69. Antetele de extensie IPv6.

Unele dintre antete au un format fix; altele conțin un număr variabil de câmpuri de lungime variabilă. Pentru acestea, fiecare element este codificat ca un tuplu (Tip, Lungime, Valoare). *Tipul* este

un câmp de 1 octet care spune ce opțiune este aceasta. Valorile *Tipului* au fost alese astfel, încât primii 2 biți spun ruterelor care nu știu cum să proceseze opțiunea ce anume să facă. Variantele sunt: sărirea opțiunii; eliminarea pachetului; eliminarea pachetului și trimiterea înapoi a unui pachet ICMP; la fel ca mai înainte, doar că nu se trimit pachete ICMP pentru adrese de trimitere multiplă (pentru a preveni ca un pachet eronat de multicast să genereze milioane de răspunsuri ICMP).

Câmpul *Lungime* este de asemenea un câmp de lungime 1 octet. El precizează cât de lungă este valoarea (de la 0 la 255). *Valoarea* este orice informație necesară, până la 255 octeți.

Antetul salt-după-salt este folosit pentru informații ce trebuie examinate de toate ruterele de pe cale. Până acum a fost definită o opțiune: suportul pentru datagrame ce depășesc de 64K. Formatul acestui antet este prezentat în fig. 5-70. Atunci când este folosit, câmpul *Lungimea informației utile* din antetul fix este zero.

Antetul următor	0	194	4
Lungimea informației utile foarte mari			

Fig. 5-70. Antetul de extensie salt-după-salt pentru datagrame mari (jumbograme).

Ca toate antetele de extensie, acesta începe cu un octet care spune ce tip de antet este următorul. Acest octet este urmat de unul care spune cât de lung este antetul salt-după-salt în octeți, excluzând primii 8 octeți, care sunt obligatorii. Toate extensiile încep la fel.

Următorii 2 octeți arată că această opțiune definește dimensiunea datagramei (codul 194) ca un număr de 4 octeți. Ultimii 4 octeți dau dimensiunea datagramei. Dimensiunile mai mici de 65.536 nu sunt permise și au ca rezultat eliminarea pachetului de către primul ruter, care va trimite înapoi un mesaj ICMP de eroare. Datagramele care folosesc acest antet de extensie sunt numite **jumbograme**. Folosirea jumbogramelor este importantă pentru aplicațiile supercalculatoarelor care trebuie să transfere gigaocteți de date eficient prin intermediul Internet-ului.

Antetul opțiunilor pentru destinație este prevăzut pentru câmpuri care trebuie interpretate numai de către gazda destinație. În versiunea inițială a IPv6 singura opțiune definită este cea de completare a acestui antet până la un multiplu de 8 octeți, așa că pentru început nu va fi folosit. El a fost inclus pentru a asigura posibilitatea ca un nou software al ruterului sau al gazdei să îl poate trata, în cazul în care cineva, cândva, se gândește la o opțiune pentru destinație.

Antetul de dirijare enumeră unul sau mai multe rutere care trebuie să fie vizitate pe calea spre destinație. Este foarte asemănător cu dirijarea aproximativă din IPv4 prin aceea că toate adresele listate trebuie vizitate în ordine, dar între acestea pot fi vizitate alte rutere nelistate. Formatul antetului de dirijare este prezentat în fig. 5-71.

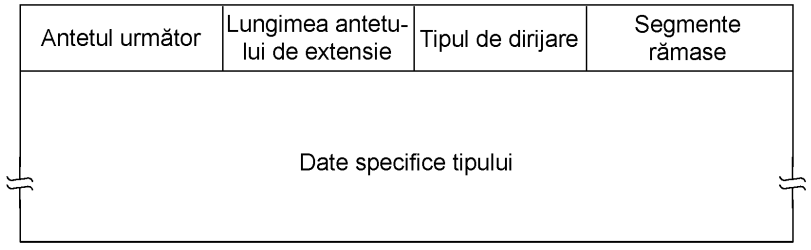


Fig. 5-71. Antetul de extensie pentru dirijare.

Primii 4 octeți ai antetului de extensie de dirijare conțin patru întregi de 1 octet. Câmpurile *Antet următor* și *Lungimea antetului* extensie au fost descrise anterior. Câmpul *Tipul dirijării* precizează formatul părții rămase din header. Tipul 0 arată că după primul cuvânt urmează un cuvânt rezervat pe 32 de biți, urmat de un număr de adrese IPv6. În viitor, în funcție de necesități, ar putea fi inventate alte tipuri. În final, câmpul *Segmente rămase* reține câte adrese din listă nu au fost vizitate și este decrementat de fiecare dată când este vizitată o adresă. Atunci când se ajunge la 0 pachetul este pe cont propriu, fără nici o indicație referitoare la ruta de urmat. De obicei, în acest punct este atât de aproape de destinație încât calea cea mai bună este evidentă.

Antetul fragment tratează fragmentarea într-un mod similar cu cel al IPv4. Antetul menține identificatorul datagramei, numărul de fragment și un bit care spune dacă mai urmează fragmente. În IPv6, spre deosebire de IPv4, numai gazda sursă poate fragmenta un pachet. Ruterele de pe cale nu pot face acest lucru. Deși această schimbare este o rupere filozofică majoră cu trecutul, ea simplifică munca ruterelor și permite ca dirijarea să se facă mai rapid. Așa cum s-a menționat mai sus, dacă un ruter este confruntat cu un pachet care este prea mare, el elimină pachetul și trimite un pachet ICMP înapoi la sursă. Această informație îi permite gazdei sursă să fragmenteze pachetul în bucăți mai mici folosind acest antet și apoi să reîncerce.

Antetul de autentificare oferă un mecanism prin care receptorul unui mesaj poate fi sigur de cel care l-a trimis. Informația utilă de siguranță criptată face posibilă criptarea conținutului unui pachet, astfel încât doar receptorul căruia îi este destinat poate să-l citească. Pentru a-și realiza misiunea aceste antete folosesc tehnici criptografice.

Controverse

Dat fiind procesul de proiectare deschisă și opiniile ferm susținute ale multora dintre persoanele implicate, nu ar trebui să fie o surpriză că multe din deciziile luate pentru IPv6 au fost foarte controversate. În cele ce urmează vom rezuma câteva dintre acestea. Pentru toate amănuntele tăioase, vezi RFC-urile.

Am menționat deja disputa legată de lungimea adresei. Rezultatul a fost un compromis: adrese de lungime fixă de 16 octeți.

O altă dispută s-a dus în jurul lungimii câmpului *Limita salturilor*. O tabără a simțit că limitarea numărului maxim de salturi la 255 (implicită în cazul folosirii unui câmp de 8 biți) ar fi o greșeală grosolană. Până la urmă, căi de 32 de salturi sunt obișnuite în zilele noastre, iar peste 10 ani pot fi obișnuite căi mult mai lungi. Aceste persoane au argumentat că folosirea unui spațiu de adrese enorm a fost clarviziune, dar folosirea unui contor minuscul de salturi a fost miopie. După părerea lor, cel mai mare păcat pe care îl poate comite un informatician este să ofere prea puțini biți într-un loc.

Răspunsul a fost că pot fi aduse argumente pentru lărgirea fiecărui câmp, conducând la un antet umflat. De asemenea, funcția câmpului *Limita salturilor* este de a împiedica hoinăreala pachetelor pentru un timp îndelungat și 65.536 de salturi este mult prea mult. În cele din urmă, pe măsură ce Internet-ul crește, se vor construi din ce în ce mai multe legături de mare distanță, făcând posibilă ajungerea dintr-o țară în alta în cel mult o jumătate de duzină de salturi. Dacă este nevoie de mai mult de 125 de salturi pentru a ajunge de la sursă sau destinație la porțile lor internaționale, ceva este în neregulă cu coloanele vertebrale naționale. Adepții celor 8 biți au câștigat această luptă.

O altă problemă spinoasă a fost dimensiunea maximă a pachetului. Comunitatea supercalculatoarelor a dorit pachete mai mari de 64 KB. Când un supercalculator începe să transfere, aceasta înseamnă într-adevăr lucru serios și nu dorește să fie întrerupt după fiecare 64KB. Argumentul împotri-

va pachetelor mari este că dacă un pachet de 1 MB ajunge la o linie T1 de 1.5 Mbps, acel pachet va monopoliza linia pentru mai mult de 5 secunde, producând o întârziere semnificativă pentru utilizatorii interactivi care partajează linia. Aici s-a ajuns la un compromis: pachetele normale au fost limitate la 64 KB, dar antetul de extensie salt-după-salt poate fi folosit pentru a permite jumbograme.

Un al treilea punct fierbinte a fost eliminarea sumei de control IPv4. Unele persoane au asimilat această mutare cu eliminarea frânelor de la mașină. Făcând acest lucru, mașina devine mai ușoară, astfel încât poate merge mai repede, dar dacă intervine ceva neașteptat, apar probleme.

Argumentul împotriva sumei de control a fost că orice aplicație care are într-adevăr grijă de integritatea datelor trebuie oricum să aibă o sumă de control la nivelul transport, așa încât menținerea a încă o sumă în IP (în plus față de suma de control a nivelului legătură de date) este un exces. Mai mult, experiența a arătat că în IPv4 calculul sumei de control IP era foarte costisitoare. Tabăra împotriva sumei de control a învins de această dată, deci IPv6 nu are o sumă de control.

Gazdele mobile au fost de asemenea un punct de conflict. Dacă un calculator portabil face jumătate din ocolul lumii, poate continua el să opereze la destinație cu aceeași adresă IPv6 sau trebuie să folosească o schemă cu agenți locali și agenți pentru străini? De asemenea, gazdele mobile introduc asimetrii în sistemul de dirijare. Se poate foarte bine întâmpla ca un mic calculator mobil să audă semnalul puternic trimis de un ruter staționar, dar ruterul staționar nu poate auzi semnalul slab trimis de gazda mobilă. În consecință, unele persoane au dorit să includă în IPv6 suport explicit pentru gazde mobile. Acest efort a eșuat pentru că nu s-a putut ajunge la un consens pentru o propunere concretă.

Probabil că cea mai mare bătălie a fost pentru securitate. Toată lumea a fost de acord că este necesară. Războiul a fost pentru unde și cum. Mai întâi unde. Argumentul pentru plasarea la nivelul rețea este că devine un serviciu standard pe care toate aplicațiile îl pot folosi fără o planificare prealabilă. Argumentul contra este că, în general, aplicațiile cu adevărat sigure doresc cel puțin criptare capăt-la-capăt, în care aplicația sursă criptează și aplicația destinație decriptează. Altfel, utilizatorul este la mila unor implementări potențial pline de pene a nivelurilor rețea, implementări asupra cărora nu are nici un control. Răspunsul la acest argument este că aceste aplicații pot să se abțină de la folosirea facilităților de securitate IP și să-și facă treaba ele însele. Replica la aceasta este că persoanele care nu au încredere că rețeaua face treaba cum trebuie, nu doresc să plătească prețul unor implementări de IP lente, greoaie, care au această facilitate, chiar dacă este dezactivată.

Un alt aspect al disputei unde să fie pusă securitatea este legat de faptul că multe țări (dar nu toate) au legi de export severe referitoare la criptografie. Unele, notabil în Franța și Irak, reduc în mare măsură folosirea internă a criptografiei, așa încât oamenii nu pot avea secrete față de poliție. Ca rezultat, orice implementare de IP care utilizează un sistem criptografic suficient de puternic pentru a fi de mare valoare nu poate fi exportat din Statele Unite (și multe alte țări) clienților din lumea întreagă. Necesitatea menținerii a două seturi de programe, unul pentru uz intern și altul pentru export, este un fapt ce întâmpină o opoziție viguroasă din partea firmelor de calculatoare.

Un punct asupra căruia nu au existat controverse este că nimeni nu se așteaptă ca Internet-ul bazat pe IPv4 să fie închis într-o duminică dimineața și să repornească ca un Internet bazat pe IPv6 luni dimineață. În schimb, vor fi convertite „insule” izolate de IPv6, inițial comunicând prin tunele. Pe măsură ce insulele IPv6 cresc, ele vor fuziona în insule mai mari. Până la urmă, toate insulele vor fuziona și Internet-ul va fi convertit complet. Dată fiind investiția masivă în rutere IPv4 în folosință curentă, procesul de conversie va dura probabil un deceniu. Din acest motiv s-a depus o enormă cantitate de efort pentru a asigura că această tranziție va fi cât mai puțin dureroasă posibil. Pentru mai multe informații despre IPv6, vezi (Loshin, 1999).

5.7 REZUMAT

Nivelul rețea furnizează servicii nivelului transport. El se poate baza fie pe circuite virtuale, fie pe datagrame. În ambele cazuri, principala sa sarcină este dirijarea pachetelor de la sursă la destinație. În subrețelele bazate pe circuite virtuale, decizia de dirijare se ia atunci când este stabilit circuitul. În subrețelele bazate pe datagrame decizia este luată pentru fiecare pachet.

În rețelele de calculatoare sunt folosiți mulți algoritmi de dirijare. Algoritmii statici includ dirijarea pe calea cea mai scurtă și inundarea. Algoritmii dinamici include dirijarea după vectorul distanțelor și dirijarea după starea legăturii. Majoritatea rețelelor actuale folosesc unul dintre acești algoritmi. Alte teme importante referitoare la dirijare sunt dirijarea ierarhică, dirijarea pentru gazde mobile, dirijarea pentru difuzare, dirijarea multidestinație și cea în rețele punct-la-punct.

Subrețelele pot deveni cu ușurință congestionate, mărind întârzierea și micșorând productivitatea pentru pachete. Proiectanții rețelelor încearcă să evite congestia printr-o proiectare adecvată. Tehnicile includ politica de retransmitere, folosirea memoriei ascunse, controlul fluxului și altele. Dacă apare congestia, ea trebuie să fie tratată. Pot fi trimise înapoi pachete de șoc, încărcarea poate fi eliminată sau se pot aplica alte metode.

Următorul pas dincolo de simpla tratare a congestiei este încercarea de a se ajunge la calitatea promisă a serviciului. Metodele care pot fi folosite pentru aceasta includ folosirea zonelor tampon la client, modelarea traficului, rezervarea resurselor și controlul accesului. Abordările care au fost proiectate pentru o bună calitate a serviciului includ servicii integrate (ca RSVP), servicii diferențiate și MPLS.

Rețelele diferă prin multe caracteristici, așa că atunci când se conectează mai multe rețele, pot să apară probleme. Uneori problemele pot fi evitate prin trecerea prin tunel a unui pachet ce traversează o rețea ostilă, dar dacă rețeaua sursă și cea destinație diferă, această abordare eșuează. Atunci când rețele diferite au dimensiunile maxime ale pachetelor diferite, se poate produce fragmentarea.

Internet-ul posedă o mare varietate de protocoale legate de nivelul rețea. Acestea includ protocolul de transport al datelor, IP, dar și protocoalele de control ICMP, ARP și RARP și protocoalele de dirijare OSPF și BGP. Internet-ul va rămâne foarte repede fără adrese IP, așa că s-a dezvoltat o nouă versiune de IP, IPv6.

5.8 PROBLEME

1. Dați două exemple de aplicații pentru care este adecvat un serviciu orientat pe conexiune. Apoi dați două exemple pentru care un serviciu fără conexiuni este cel mai potrivit.

2. Există vreo situație în care un serviciu cu circuit virtual va (sau cel puțin ar putea) livra pachetele în altă ordine? Explicați.

3. Subrețelele bazate pe datagrame dirijează fiecare pachet ca pe o unitate separată, independentă de toate celelalte. Subrețelele bazate pe circuite virtuale nu trebuie să facă acest lucru, pentru că fiecare pachet de date urmează o cale predeterminată. Oare această observație înseamnă că

subrețelele bazate pe circuite virtuale nu au nevoie de capacitatea de a dirija pachetele izolate de la o sursă arbitrară către o destinație arbitrară? Explicați răspunsul dat.

4. Dați trei exemple de parametri ai protocolului care ar putea fi negociați atunci când este inițiată o conexiune.

5. Considerați următoarea problemă de proiectare, privind implementarea unui serviciu cu circuit virtual. Dacă în interiorul unei subrețele sunt folosite circuite virtuale, fiecare pachet de date trebuie să conțină un antet de 3 octeți, iar fiecare ruter trebuie să aloce 8 octeți de memorie pentru identificarea circuitelor. Dacă intern sunt folosite datagrame, sunt necesare antete de 15 octeți, dar nu este nevoie de spațiu pentru tabela ruterului. Capacitatea de transmisie costă 1 cent per 10^6 octeți, per salt. Memoria foarte rapidă pentru ruter poate fi cumpărată la prețul de 1 cent per octet și se depreciază peste doi ani (considerând numai orele de funcționare). Din punct de vedere statistic, o sesiune medie durează 1000 de secunde, iar în acest timp sunt transmise 200 de pachete. Un pachet mediu are nevoie de patru salturi. Care implementare este mai ieftină și cu cât?

6. Presupunând că toate ruterele și gazdele funcționează normal și că întregul software din rutere și gazde nu conține nici o eroare, există vreo șansă, oricât de mică, ca un pachet să fie livrat unei destinații greșite?

7. Considerați rețeaua din fig. 5-7, dar ignorați ponderile de pe linii. Presupuneți că algoritmul de rutare utilizat este cel de inundare. Listați toate rutele pe care le va parcurge un pachet trimis de la *A* la *D,* al cărui număr maxim de salturi este 3. De asemenea precizați câte noduri consumă inutil bandă de transmisie.

8. Formulați o euristică simplă pentru găsirea a două căi de la o sursă dată la o destinație dată care pot supraviețui pierderii oricărei linii de comunicație (presupunând că există două astfel de căi). Ruterele sunt considerate suficient de fiabile, deci nu este necesar să ne îngrijoreze posibilitatea căderii ruterelor.

9. Considerați subrețeaua din fig. 5-13(a). Se folosește dirijarea după vectorul distanțelor și următorii vectori tocmai au sosit la ruterul C: de la B: (5, 0, 8, 12, 6, 2); de la D: (16, 12, 6, 0, 9, 10); și de la E: (7, 6, 3, 9, 0, 4). Întârzierile măsurate către B, D și E, sunt 6, 3 și respectiv 5. Care este noua tabelă de dirijare a lui C? Precizați atât linia de ieșire folosită, cât și întârzierea presupusă.

10. Dacă întârzierile sunt înregistrate ca numere de 8 octeți într-o rețea cu 50 de rutere și vectorii cu întârzieri sunt schimbați de două ori pe secundă, cât din lărgimea de bandă a unei linii (duplex integral) este consumată de algoritmul distribuit de dirijare? Presupuneți că fiecare ruter are trei linii către alte rutere.

11. În fig. 5-14 rezultatul operației SAU logic a celor două mulțimi de biți ACF este 111 în fiecare linie. Este acesta doar un accident întâmplat aici sau este valabil pentru toate subrețelele, în toate împrejurările?

12. La dirijarea ierarhică cu 4800 de rutere, ce dimensiuni ar trebui alese pentru regiune și grup, astfel încât să se minimizeze dimensiunea tabelei de dirijare pentru o ierarhie cu trei niveluri?

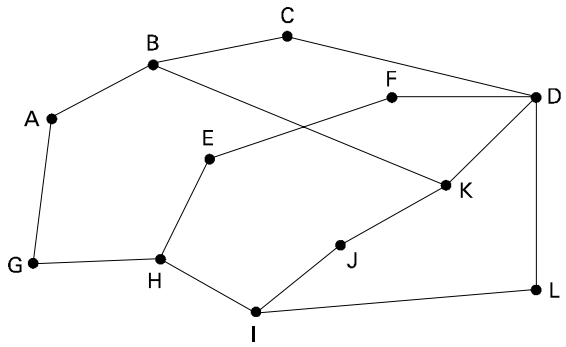
Un punct de pornire este ipoteza că o soluție cu k clustere de k regiuni de k rutere este aproape de optim, ceea ce înseamnă că valoarea k este aproximativ rădăcina cubică a lui 4800 (aproximativ 16). Folosiți încercări repetate pentru a verifica combinațiile cu toți cei trei parametri în vecinătatea lui 16.

13. În text s-a afirmat că atunci când un sistem gazdă mobil nu este acasă, pachetele trimise către LAN-ul de domiciliu sunt interceptate de agentul său local. Pentru o rețea IP pe un LAN 802.3, cum va realiza agentul local această interceptare?

14. Privind subrețeaua din fig. 5-6, câte pachete sunt generate de o difuzare de la B, folosind:
a) urmărirea căii inverse?
b) arborele de scufundare?

15. Fie rețeaua din fig. 5-16(a). Să ne imaginăm că între F și G este adăugată o nouă linie, dar arborele de scufundare din fig. 5-16(b) rămâne neschimbat. Ce modificări survin în fig. 5-16(c) ?

16. Calculați un arbore de acoperire pentru trimitere multiplă pentru ruterul C din rețeaua de mai jos pentru un grup cu membrii la ruterele A, B, C, D, E, F, I și K.



17. În fig. 5-20, difuzează vreodată nodurile H și I la căutarea pornită din A ?

18. Să presupunem că nodul B din fig. 5-20 tocmai a pornit și nu are nici o informație de dirijare în tabelele sale. Brusc, are nevoie de o cale către H. El va difuza pachete cu *TTL* setat la 1, 2, 3 și așa mai departe. De câte runde are nevoie pentru a găsi o cale ?

19. În cea mai simplă variantă a algoritmului Chord pentru căutarea punct-la-punct, căutările nu folosesc tabela de indicatori. În loc de aceasta, ele sunt lineare în jurul cercului în oricare direcție. Poate un nod determina cu precizie în ce direcție trebuie să caute ? Discutați răspunsul.

20. Fie cercul Chord din fig. 5-24. Să presupunem că nodul 10 pornește brusc. Afectează aceasta tabela de indicatori a nodului 1, și dacă da, cum ?

21. Ca un posibil mecanism de control al congestiei într-o subrețea ce folosește intern circuite virtuale, un ruter poate amâna confirmarea unui pachet primit până când (1) știe că ultima sa transmisie de-a lungul circuitului virtual a fost primită cu succes și (2) are un tampon liber. Pentru simplitate, să presupunem că ruterele utilizează un protocol stop-and-wait (pas-cu-pas) și că fie-

care circuit virtual are un tampon dedicat pentru fiecare direcție a traficului. Dacă este nevoie de T sec pentru a trimite un pachet (date sau confirmare) și sunt n rutere de-a lungul căii, care este viteza cu care pachetele sunt livrate gazdei destinație? Presupunem că erorile de transmisie sunt rare, iar conexiunea gazdă-ruter este infinit de rapidă.

22. O subrețea de tip datagramă permite ruterelor să elimine pachete de câte ori este necesar. Probabilitatea ca un ruter să renunțe la un pachet este p. Considerăm cazul unei gazde sursă conectate cu un ruter sursă, care este conectat cu un ruter destinație și apoi cu gazda destinație. Dacă unul dintre rutere elimină un pachet, până la urmă gazda sursă va depăși limita de timp și va încerca din nou. Dacă liniile gazdă-ruter și ruter-ruter sunt ambele numărate ca salturi, care este numărul mediu de:
a) salturi per transmisie pe care le face un pachet?
b) transmisii determinate de un pachet?
c) salturi necesare pentru un pachet primit?

23. Descrieți două diferențe majore dintre metoda bitului de avertizare și metoda RED.

24. Dați o explicație pentru faptul că algoritmul găleții găurite permite un singur pachet per tact, indiferent de cât de mare este pachetul.

25. Într-un sistem oarecare este utilizată varianta cu numărarea octeților a algoritmului găleții găurite. Regula este că pot fi trimise la fiecare tact un pachet de 1024 de octeți, două pachete de 512 octeți etc. Formulați o limitare serioasă a acestui sistem care nu a fost menționată în text.

26. O rețea ATM utilizează pentru modelarea traficului o schemă de tip găleată cu jetoane (token bucket). La fiecare 5 μsec în găleată este introdus un nou jeton. Fiecare jeton este asociat unei singure celule, care conține 48 octeți de date. Care este viteza maximă a datelor care poate fi asigurată?

27. Un calculator dintr-o rețea de 6 Mbps este guvernat de o schemă de tip găleată cu jetoane. Aceasta se umple cu viteza de 1 Mbps. Ea este umplută inițial la capacitatea maximă, cu 8 megabiți. Cât timp poate calculatorul să transmită cu întreaga viteză de 6 Mbps?

28. Să ne imaginăm o specificație de flux care are dimensiunea maximă a pachetului de 1000 de octeți, viteza găleții cu jetoane de 10 milioane de octeți/sec, capacitatea găleții de 1 milion de octeți și viteza maximă de transmisie de 50 de milioane de octeți/sec. Cât timp poate dura o rafală la viteza maximă?

29. Rețeaua din fig. 5-37 folosește RSVP cu arbori multidestinație pentru gazdele 1 și 2, după cum este ilustrat. Să presupunem că gazda 3 cere un canal cu lățimea de bandă de 2MB/sec pentru un flux de la gazda 1 și alt canal cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. În același timp gazda 4 cere un canal cu lărgimea de bandă de 2MB/sec pentru un flux de la gazda 1 și gazda 5 cere un alt cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. Ce lărgime de bandă totală va fi rezervată la ruterele A, B, C, E, H, J, K, L pentru aceste cereri?

30. Procesorul dintr-un ruter poate prelucra 2 milioane de pachete/sec. Încărcarea oferită lui este de 1,5 milioane pachete/sec. Dacă o rută de la sursă la destinație trece prin 10 rutere, cât timp se consumă în așteptare și pentru servirea de către procesoare?

31. Fie utilizatorul unor servicii diferențiate cu rutare expeditivă. Există o garanție că pachetele prioritare vor suferi o întârziere mai mică decât pachetele normale? De ce sau de ce nu?

32. Este nevoie de fragmentare în rețele concatenate bazate pe circuite virtuale sau numai în sisteme cu datagrame?

33. Trecerea prin tunel printr-o subrețea de circuite virtuale concatenate este simplă: ruterul multiprotocol de la un capăt stabilește circuitul virtual către celălalt capăt și trece pachetele prin el. Poate această trecere prin tunel să fie folosită și în subrețelele bazate pe datagrame? Dacă da, cum?

34. Să presupunem că gazda *A* este conectată la ruterul *R1*, *R1* este conectat la alt ruter *R2*, și *R2* este conectat la gazda *B*. Să presupunem că un mesaj TCP care conține 900 octeți de date și 20 de octeți de antet TCP este transmis codului IP aflat pe gazda *A* pentru a fi transmis lui *B*. Arătați câmpurile *Lungimea totală, Identificare*, *DF* ,*MF* și *Deplasamentul fragmentului* din antetul IP din fiecare pachet transmis prin cele trei legături. Se presupune că legătura *A-R1* poate suporta o lungime maximă de cadru de 1024 de octeți incluzând un antet de cadru de 14 octeți, legătura *R1-R2* poate suporta o lungime maximă de cadru de 512 de octeți incluzând un antet de cadru de 8 octeți și legătura *R2-B* poate suporta o lungime maximă de cadru de 512 octeți incluzând un antet de cadru de 12 octeți.

35. Un ruter distruge pachetele IP a căror lungime totală (date plus antet) este de 1024 octeți. Presupunând că pachetele trăiesc pentru 10 sec, care este viteza maximă a liniei la care poate opera ruterul fără a fi în pericol să cicleze prin spațiul numerelor de ID al datagramelor IP.

36. O datagramă IP care folosește opțiunea *Dirijare strictă de la sursă* trebuie să fie fragmentată. Credeți că opțiunea este copiată în fiecare fragment, sau este suficient să fie pusă numai în primul fragment? Explicați răspunsul.

37. Să presupunem că pentru o adresă de clasă B, partea care specifică rețeaua utilizează 20 de biți în loc de 16 biți. Câte rețele de clasă B se pot obține?

38. Transformați adresa IP a cărei reprezentare zecimală este C22F1582 într-o notație zecimală cu puncte.

39. O rețea din Internet are masca de subrețea 255.255.240.0. Care este numărul maxim de gazde din subrețea?

40. Un număr mare de adrese IP consecutive sunt disponibile începând cu 198.16.0.0. Să presupunem că patru organizații, *A*, *B*, *C*, *D*, cer câte 4000, 2000, 4000 și 8000 adrese, în această ordine. Precizați, pentru fiecare dintre ele, prima și ultima adresă IP atribuită, precum și masca în notația w.x.y.z/s.

41. Un ruter tocmai a primit următoarele noi adrese IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 și 57.6.129.0/21. Dacă toate folosesc aceeași linie de ieșire, pot fi ele compuse? Dacă da, ce va rezulta? Dacă nu, de ce nu ?

42. Setul de adrese IP de la 29.18.0.0 la 19.18.128.255 au fost reunite la 29.18.9.9/17. Totuși există un spațiu de 1024 de adrese nealocate, de la 29.18.60.0 la 29.18.63.255, care sunt brusc alocate

unei gazde care folosește altă linie de ieșire. Este nevoie acum ca adresa agregată să fie spartă în blocurile constituente, să se adauge blocurile noi la tabelă și să se vadă apoi dacă este posibilă o altă reunire? Dacă nu, ce se poate face ?

43. Un ruter are următoarele intrări (CIDR) în tabela sa de dirijare :

Adresă/mască	**Următorul salt**
135.46.56.0/22	Interfața 0
135.46.60.0/22	Interfața 1
192.53.40.0/23	Ruter 1
Implicit	Ruter 2

Pentru fiecare dintre următoarele adrese IP, ce face ruterul dacă primește un pachet cu respectiva adresă?

a) 135.46.63.10
b) 135.46.57.14
c) 135.46.52.2
d) 192.53.40.7
e) 192.53.56.7

44. Multe companii au politica de a avea două (sau mai multe) rutere conectate la Internet pentru a avea redundanță în caz că unul dintre ele nu mai funcționează. Mai este această politică posibilă cu NAT ? Explicați răspunsul.

45. Tocmai i-ați explicat unui prieten protocolul ARP. Când ați terminat, el spune: „Am înțeles. ARP oferă un serviciu nivelului rețea, deci face parte din nivelul legăturii de date." Ce îi veți spune?

46. Atât ARP cât și RARP realizează corespondența adreselor dintr-un spațiu în altul. Din acest punct de vedere cele două protocoale sunt similare. Totuși, implementările lor sunt fundamental diferite. Care este diferența esențială dintre ele?

47. Descrieți un procedeu pentru reasamblarea fragmentelor IP la destinație.

48. Cei mai mulți algoritmi de reasamblare a datagramelor IP au un ceas pentru a evita ca un fragment pierdut să țină ocupate pentru totdeauna tampoanele de reasamblare. Să presupunem că o datagramă este împărțită în patru fragmente. Primele trei sosesc, dar ultimul este întârziat. În cele din urmă timpul expiră și cele trei fragmente sunt eliminate din memoria receptorului. Puțin mai târziu, sosește și ultimul fragment. Ce ar trebui făcut cu el?

49. Atât la IP cât și la ATM, suma de control acoperă numai antetul, nu și datele. De ce credeți că s-a ales această soluție?

50. O persoană care locuiește în Boston călătorește la Minneapolis, luându-și calculatorul portabil cu sine. Spre surprinderea sa, LAN-ul de la destinația din Minneapolis este un LAN IP fără fir, deci nu trebuie să se conecteze. Este oare necesar să se recurgă la întreaga poveste cu agenți locali și agenți străini pentru ca mesajele de poștă electronică și alte tipuri de trafic să-i parvină corect?

51. IPv6 folosește adrese de 16 octeți. Dacă la fiecare picosecundă este alocat câte un bloc de 1 milion de adrese, cât timp vor exista adrese disponibile?

52. Câmpul *Protocol* folosit în antetul IPv4 nu este prezent în antetul fix pentru IPv6. De ce?

53. Când se introduce protocolul IPv6, protocolul ARP trebuie să fie modificat? Dacă da, modificările sunt conceptuale sau tehnice?

54. Scrieți un program care să simuleze dirijarea prin inundare. Fiecare pachet ar conține un contor care este decrementat la fiecare salt. Când contorul ajunge la zero, pachetul este eliminat. Timpul este discret, iar fiecare linie manevrează un pachet într-un interval de timp. Realizați trei versiuni ale acestui program: toate liniile sunt inundate, sunt inundate toate liniile cu excepția liniei de intrare, sau sunt inundate numai cele mai bune k linii (alese statistic). Comparați inundarea cu dirijarea deterministă ($k = 1$) în termenii întârzierii și lărgimii de bandă folosite.

55. Scrieți un program care simulează o rețea de calculatoare ce folosește un timp discret. Primul pachet din coada de așteptare a fiecărui ruter face un salt per interval de timp. Fiecare ruter are numai un număr finit de zone tampon. Dacă un pachet sosește și nu este loc pentru el, el este eliminat și nu mai este retransmis. În schimb, există un protocol capăt-la-capăt, complet, cu limită de timp și pachete de confirmare, care va regenera în cele din urmă pachetul de la ruterul sursă. Reprezentați grafic productivitatea rețelei ca funcție de limita de timp, parametrizată de rata erorilor.

56. Scrieți o funcție pentru retransmiterea într-un ruter IP. Procedura are un parametru, o adresă IP. De asemenea are acces la o tabelă globală constând dintr-un vector de tripleți. Fiecare triplet conține trei întregi: o adresă IP, o mască de subrețea și linia de ieșire ce trebuie folosită. Funcția caută adresa IP în tabelă folosind CIDR și întoarce ca valoare linia ce trebuie folosită.

57. Folosiți programele *traceroute* (UNIX) sau *tracert* (Windows) pentru a urmări calea de la calculatorul personal până la diverse universități de pe alte continente. Creați o listă a legăturilor transoceanice pe care le-ați descoperit. Câteva sit-uri de încercat sunt: www.berkely.edu (California), www.u-tokyo.ac.jp (Tokyo), www.mit.edu (Massachusetts), www.vu.nl (Amsterdam), www.usyd.edu.au (Sydney), www.ucl.ac.uk (Londra), www.uct.ac.za (Cape Town).

6

NIVELUL TRANSPORT

Nivelul transport nu este doar un alt nivel, el este miezul întregii ierarhii de protocoale. Sarcina sa este de a transporta date de la mașina sursă la mașina destinație într-o manieră sigură și eficace din punctul de vedere al costurilor, independent de rețeaua sau rețelele fizice utilizate. Fără nivelul transport și-ar pierde sensul întregul concept de ierarhie de protocoale. În acest capitol vom studia în detaliu nivelul transport, incluzând serviciile, arhitectura, protocoalele și performanțele sale.

6.1 SERVICIILE OFERITE DE NIVELUL TRANSPORT

În secțiunile următoare vom face o prezentare a serviciilor oferite de nivelul transport. Vom studia serviciile oferite nivelului aplicație. Pentru a face problema serviciului de transport mai concretă, vom examina două seturi de primitive ale nivelului transport. La început ne vom ocupa de unul simplu (dar ipotetic), pentru a arăta ideile de bază. Apoi va fi studiată interfața folosită în mod obișnuit în Internet.

6.1.1 Servicii furnizate nivelurilor superioare

Scopul principal al nivelului transport este de a oferi servicii eficiente, sigure și ieftine utilizatorilor, în mod normal procese aparținând nivelului aplicație. Pentru a atinge acest scop, nivelul transport utilizează serviciile oferite de nivelul rețea. Hardware-ul și/sau software-ul care se ocupă de toa-

te acestea în cadrul nivelului transport poartă numele de **entitate de transport**. Entitatea de transport poate aparține nucleului sistemului de operare, unui proces distinct, unei biblioteci legate de aplicațiile de rețea sau poate fi găsită în cadrul plăcii de rețea. Relația (logică) între nivelurile rețea, transport și aplicație este prezentată în fig. 6-1.

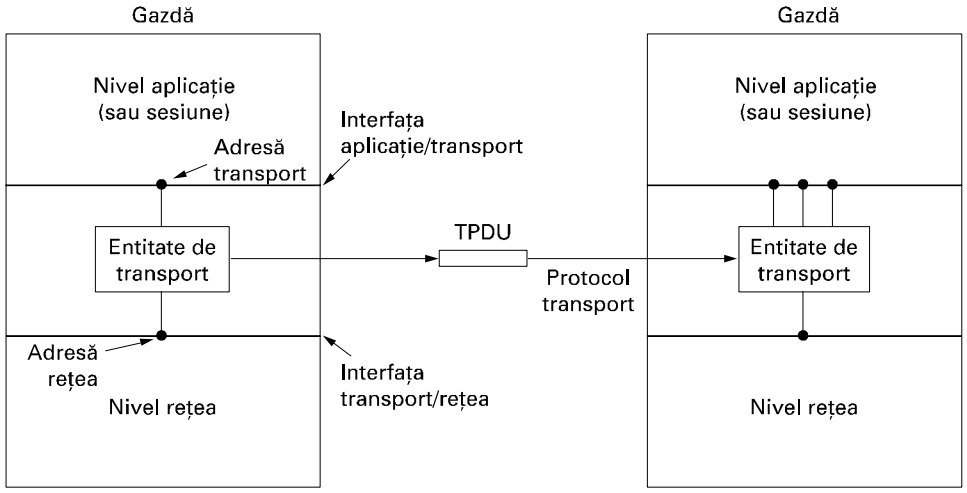


Fig. 6-1. Nivelurile rețea, transport și aplicație.

Cele două tipuri de servicii: orientate pe conexiune sau datagramă, existente în cadrul nivelului rețea, se regăsesc și la acest nivel. Serviciul orientat pe conexiune de la nivelul transport are multe asemănări cu cel de la nivel rețea. În ambele cazuri, conexiunile au trei faze: stabilirea conexiunii, transferul de date și eliberarea conexiunii. Adresarea și controlul fluxului sunt și ele similare pentru ambele niveluri. Mai mult, chiar și serviciul fără conexiune al nivelului transport este foarte asemănător cu cel al nivelului rețea.

O întrebare evidentă este atunci: dacă serviciile la nivel transport sunt atât de asemănătoare cu cele de la nivel rețea, de ce este nevoie de două niveluri distincte? De ce nu este suficient un singur nivel? Răspunsul este unul subtil, dar extrem de important, și ne cere să ne întoarcem la fig. 1-9. Codul pentru nivelul transport este executat în întregime pe mașinile utilizatorilor, dar nivelul rețea este executat în cea mai mare parte de mediul de transport (cel puțin pentru rețelele larg răspândite geografic - WAN). Ce s-ar întâmpla dacă nivelul rețea ar oferi servicii neadecvate? Dar dacă acesta ar pierde frecvent pachete? Ce se întâmplă dacă din când în când ruterul cade?

Ei bine, în toate aceste cazuri apar probleme. Deoarece utilizatorii nu pot controla nivelul rețea, ei nu pot rezolva problema unor servicii de proastă calitate folosind rutere mai bune sau adăugând o tratare a erorilor mai sofisticată la nivelul legătură de date. Singura posibilitate este de a pune deasupra nivelului rețea un alt nivel care să amelioreze calitatea serviciilor. Dacă pe o subrețea orientată pe conexiune, o entitate de transport este informată la jumătatea transmisiei că a fost închisă abrupt conexiunea sa la nivel rețea, fără nici o indicație despre ceea ce s-a întâmplat cu datele aflate în acel moment în tranzit, ea poate iniția o altă conexiune la nivel rețea cu entitatea de transport aflată la distanță. Folosind această nouă conexiune, ea își poate întreba corespondenta care date au ajuns la destinație și care nu, și poate continua comunicarea din locul de unde a fost întreruptă.

În esență, existența nivelului transport face posibil ca serviciile de transport să fie mai sigure decât cele echivalente de la nivelul rețea. Pachetele pierdute sau incorecte pot fi detectate și corectate de către nivelul transport. Mai mult, primitivele serviciului de transport pot fi implementate ca apeluri către procedurile de bibliotecă, astfel încât să fie independente de primitivele de la nivelul rețea. Apelurile nivelului rețea pot să varieze considerabil de la o rețea la alta (de exemplu, serviciile fără conexiune într-o rețea locală pot fi foarte diferite de serviciile orientate pe conexiune dintr-o rețea larg răspândită geografic). Ascunzând serviciul rețea în spatele unui set de primitive ale serviciului transport, schimbarea serviciului rețea necesită numai înlocuirea unui set de proceduri de bibliotecă cu un altul care face același lucru, cu un serviciu inferior diferit.

Mulțumită nivelului transport, programatorii de aplicații pot scrie cod conform unui set standard de primitive, pentru a rula pe o mare varietate de rețele, fără să își pună problema interfețelor de subrețea diferite sau transmisiilor nesigure. Dacă toate rețelele reale ar fi perfecte și toate ar avea același set de primitive și ar fi garantate să nu se schimbe niciodată, atunci probabil că nivelul transport nu ar mai fi fost necesar. Totuși, în lumea reală el îndeplinește importanta funcție de a izola nivelurile superioare de tehnologia, arhitectura și imperfecțiunile subrețelei.

Din această cauză, în general se poate face o distincție între nivelurile de la 1 la 4, pe de o parte, și cel (cele) de deasupra, pe de altă parte. Primele pot fi văzute ca **furnizoare de servicii de transport,** iar ultimele ca **utilizatoare de servicii de transport**. Această distincție între utilizatori și furnizori are un impact considerabil în ceea ce privește proiectarea arhitecturii de niveluri și conferă nivelului transport o poziție cheie, acesta fiind limita între furnizorul și utilizatorul serviciilor sigure de transmisie de date.

6.1.2 Primitivele serviciilor de transport

Pentru a permite utilizatorului să acceseze serviciile de transport, nivelul transport trebuie să ofere unele operații programelor aplicație, adică o interfață a serviciului transport. Fiecare serviciu de transport are interfața sa. În acest capitol, vom examina mai întâi un serviciu de transport simplu (ipotetic) și interfața sa pentru a vedea aspectele esențiale. În secțiunea următoare vom analiza un exemplu real.

Serviciul transport este similar cu cel rețea, dar există și câteva diferențe importante. Principala diferență este că serviciul rețea a fost conceput pentru a modela serviciile oferite de rețelele reale. Acestea pot pierde pachete, deci serviciile la nivel rețea sunt în general nesigure.

În schimb, serviciile de transport (orientate pe conexiune) sunt sigure. Desigur, în rețelele reale apar erori, dar este tocmai acesta este scopul nivelului transport: să furnizeze un serviciu sigur deasupra unui nivel rețea nesigur.

Ca exemplu, să considerăm două procese conectate prin 'pipe'-uri (tuburi) în UNIX. Acestea presupun o conexiune perfectă între ele. Ele nu vor să aibă de-a face cu confirmări, pachete pierdute, congestii sau altele asemănătoare. Ele au nevoie de o conexiune sigură în proporție de 100%. Procesul A pune datele la un capăt al tubului, iar procesul B le ia de la celălalt capăt. Aceasta este exact ceea ce face un serviciu transport orientat pe conexiune: ascunde imperfecțiunile rețelei, astfel încât procesele utilizator pot să presupună existența unui flux de date fără erori.

În același timp nivelul transport furnizează și un serviciu nesigur. Totuși, sunt puține de spus în legătură cu acesta, așa că în acest capitol ne vom concentra atenția asupra serviciului orientat pe conexiune. Cu toate acestea, există unele aplicații, cum sunt programele client-server și fluxurile multimedia, care beneficiază de transport fără conexiune, deci vom vorbi puțin despre ele mai târziu.

O a doua diferență între serviciul rețea și cel de transport se referă la destinațiile lor. Serviciul rețea este folosit doar de entitățile de transport. Puțini utilizatori scriu ei înșiși entitățile de transport și, astfel, puțini utilizatori sau programe ajung să vadă vreodată serviciile rețea așa cum sunt ele. În schimb, multe programe (și programatori) folosesc primitivele de transport. De aceea, serviciul transport trebuie să fie ușor de utilizat.

Ca să ne facem o idee despre cum poate arăta un serviciu de transport, să considerăm cele cinci primitive prezentate în fig. 6-2. Această interfață este într-adevăr simplă, dar prezintă trăsăturile de bază ale oricărei interfețe orientate pe conexiune a nivelului transport. Ea permite programelor de aplicație să stabilească, să utilizeze și să elibereze conexiuni, ceea ce este suficient pentru multe aplicații.

Primitiva	Unitatea de date trimisă	Explicații
LISTEN	(nimic)	Se blochează până când un proces încearcă să se conecteze
CONNECT	CONNECTION REQ.	Încearcă să stabilească conexiunea
SEND	DATE	Transmite informație
RECEIVE	(nimic)	Se blochează până când primește date trimise
DISCONNECT	DISCONNECTION REQ.	Trimisă de partea care vrea să se deconecteze

Fig. 6-2. Primitivele unui serviciu de transport simplu.

Pentru a vedea cum pot fi utilizate aceste primitive, să considerăm o aplicație cu un server și un număr oarecare de clienți la distanță. La început, serverul apelează primitiva LISTEN, în general prin apelul unei funcții de bibliotecă care face un apel sistem pentru a bloca serverul până la apariția unei cereri client. Atunci când un client vrea să comunice cu serverul, el va executa un apel CONNECT. Entitatea de transport tratează acest apel blocând apelantul și trimițând un pachet la server. Acest pachet încapsulează un mesaj către entitatea de transport de pe server.

Este momentul să facem câteva precizări în legătură cu terminologia. În lipsa unui termen mai bun vom folosi acronimul **TPDU** (**Transport Protocol Data Unit -** unitate de date a protocolului de transport) pentru toate mesajele schimbate între două entități de transport corespondente. Astfel, TPDU-urile (schimbate la nivelul transport) sunt conținute în pachete (utilizate de nivelul rețea). La rândul lor, pachetele sunt conținute în cadre (utilizate la nivelul legătură de date). Atunci când este primit un cadru, nivelul legătură de date prelucrează antetul cadrului și dă conținutul util nivelului rețea. Entitatea rețea prelucrează antetul pachetului și pasează conținutul util entității de transport. Această ierarhie este ilustrată în fig. 6-3.

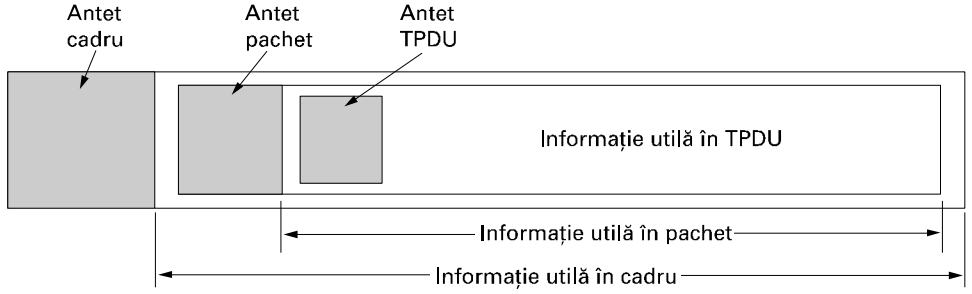


Fig. 6-3. Ierarhia de cadre, pachete și TPDU-uri.

Revenind la exemplul nostru, apelul CONNECT al clientului generează un TPDU de tip CONNECTION REQUEST care îi este trimis serverului. Atunci când acesta ajunge, entitatea de transport verifică dacă serverul este blocat într-un apel LISTEN (deci dacă așteaptă o cerere de conexiune). În acest caz, deblochează serverul și trimite înapoi clientului un TPDU CONNECTION ACCEPTED. Atunci când acest TPDU ajunge la destinație, clientul este deblocat și conexiunea este stabilită.

Acum pot fi schimbate date folosindu-se primitivele SEND și RECEIVE. Cea mai simplă posibilitate este ca una din părți să facă un apel RECEIVE (blocant) așteptând ca cealaltă parte să execute un SEND. Atunci când sosește un TPDU, receptorul este deblocat. El poate prelucra TPDU-ul și trimite o replică. Atâta vreme cât amândouă părțile știu cine este la rând să trimită mesaje și cine este la rând să recepționeze, totul merge bine.

Trebuie să observăm că la nivelul transport, chiar și un schimb de date simplu, unidirecțional, este mult mai complicat decât la nivelul rețea. Fiecare pachet de date trimis va fi (în cele din urmă) confirmat. Pachetele care conțin TPDU-uri de control sunt de asemenea confirmate, implicit sau explicit. Aceste confirmări sunt gestionate de entitățile de transport folosind protocoalele de la nivelul rețea și nu sunt vizibile utilizatorilor nivelului transport. Similar, entitățile de transport trebuie să se ocupe de ceasuri și de retransmisii. Nimic din tot acest mecanism nu este vizibil pentru utilizatorii nivelului transport, pentru care o conexiune este un tub fără pierderi: un utilizator îndeasă biți la un capăt și aceștia apar, ca prin minune, la capătul celalalt. Această capacitate de a ascunde complexitatea este motivul care face ierarhia de protocoale să fie un instrument atât de puternic.

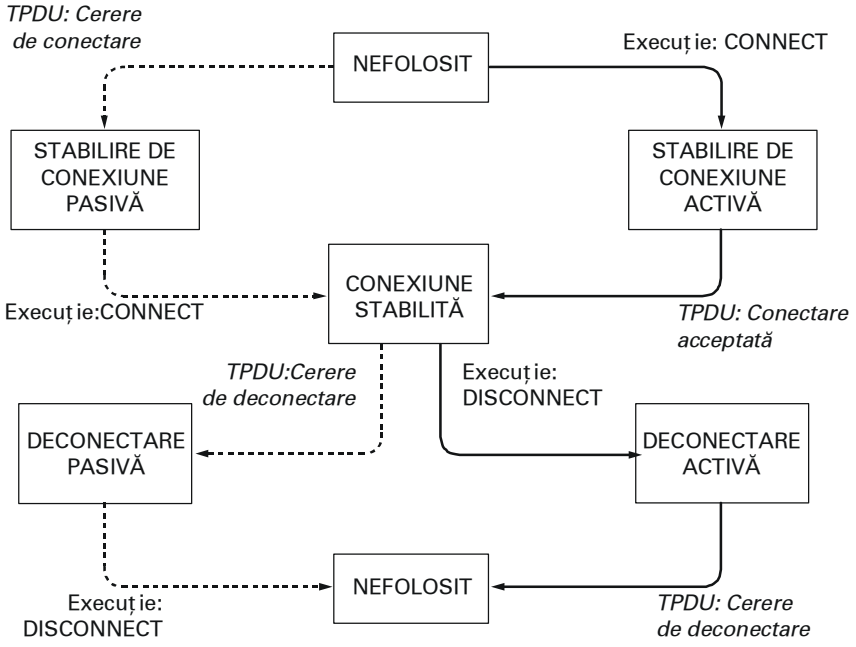


Fig. 6-4. Diagrama de stări pentru o schemă simplă de control al conexiunii. Tranzițiile etichetate cu italice sunt cauzate de sosirea unor pachete. Liniile continue indică secvența de stări a clientului. Liniile punctate indică secvența de stări a serverului.

Atunci când o conexiune nu mai este necesară, ea trebuie eliberată pentru a putea elibera și spațiul alocat în tabelele corespunzătoare din cele două entități de transport. Deconectările se pot face în două variante: asimetrică sau simetrică. În varianta asimetrică, oricare dintre utilizatori poate apela o primitivă DISCONNECT, ceea ce va avea ca rezultat trimiterea unui TPDU DISCONNECT REQUEST entității de transport aflate la distanță. La sosirea acestuia conexiunea este eliberată.

În varianta simetrică fiecare direcție este închisă separat, independent de cealaltă. Atunci când una din părți face un apel DISCONNECT, însemnând că nu mai sunt date de trimis, ea va putea încă recepționa datele transmise de entitatea de transfer aflată la distanță. În acest model conexiunea este eliberată dacă ambele părți au apelat DISCONNECT.

O diagramă de stări pentru stabilirea și eliberarea conexiunilor folosind aceste primitive simple este prezentată în fig. 6-4. Fiecare tranziție este declanșată de un eveniment: fie este executată o primitivă de către utilizatorul local al nivelului transport, fie este primit un pachet. Pentru simplitate vom presupune că fiecare TPDU este confirmat separat. Vom presupune de asemenea că este folosit un model de deconectare simetric, clientul inițiind acțiunea. Trebuie reținut că acesta este un model foarte simplu, în secțiunile următoare vom analiza modele reale.

6.1.3 Socluri Berkeley

Vom trece în revistă acum un alt set de primitive de transport: primitivele pentru socluri TCP folosite în sistemul de operare Berkeley-UNIX. Primitivele sunt enumerate în fig. 6-5. În general putem spune că acestea sunt similare modelului din capitolul precedent, dar oferă mai multe caracteristici și flexibilitate. Nu vom detalia TPDU-urile existente; această discuție mai are de așteptat până în momentul când vom studia TCP, mai târziu, în acest capitol.

Primitiva	Funcția
SOCKET	Creează un nou punct de capăt al comunicației
BIND	Atașează o adresă locală la un soclu
LISTEN	Anunță capacitatea de a accepta conexiuni; determină mărimea cozii
ACCEPT	Blochează apelantul până la sosirea unei cereri de conexiune
CONNECT	Tentativă (activă) de a stabili o conexiune
SEND	Trimite date prin conexiune
RECEIVE	Recepționează date prin conexiune
CLOSE	Eliberează conexiunea

Fig. 6-5. Primitivele pentru socluri TCP

Primele patru primitive din tabel sunt executate, în această ordine, de către server. Primitiva SOCKET creează un nou capăt al conexiunii și alocă spațiu pentru el în tabelele entității de transport. În parametrii de apel se specifică formatul de adresă utilizat, tipul de serviciu dorit (de exemplu, flux sigur de octeți) și protocolul. Un apel SOCKET reușit întoarce un descriptor de fișier (la fel ca un apel OPEN) care va fi utilizat în apelurile următoare.

Soclurile nou create nu au încă nici o adresă. Atașarea unei adrese se face utilizând primitiva BIND. Odată ce un server a atașat o adresă unui soclu, clienții se pot conecta la el. Motivul pentru care apelul SOCKET nu creează adresa direct este că unor procese le pasă de adresa lor (de exemplu, unele folosesc aceeași adresă de ani de zile și oricine cunoaște această adresă), în timp ce altele nu.

Urmează apelul LISTEN, care alocă spațiu pentru a reține apelurile primite în cazul când mai mulți clienți încearcă să se conecteze în același timp. Spre deosebire de modelul din primul nostru exemplu, aici LISTEN nu mai este un apel blocant.

Pentru a se bloca și a aștepta un apel, serverul execută o primitivă ACCEPT. Atunci când sosește un TPDU care cere o conexiune, entitatea de transport creează un nou soclu cu aceleași proprietăți ca cel inițial și întoarce un descriptor de fișier pentru acesta. Serverul poate atunci să creeze un nou proces sau fir de execuție care va gestiona conexiunea de pe noul soclu și să aștepte în continuare cereri de conexiune pe soclul inițial. ACCEPT returnează un descriptor normal de fișier, care poate fi folosit pentru citirea și scrierea în mod standard, la fel ca pentru fișiere.

Să privim acum din punctul de vedere al clientului: și în acest caz, soclul trebuie creat folosind o primitivă SOCKET, dar primitiva BIND nu mai este necesară, deoarece adresa folosită nu mai este importantă pentru server. Primitiva CONNECT blochează apelantul și demarează procesul de conectare. Când acesta s-a terminat (adică atunci când TPDU-ul corespunzător a fost primit de la server), procesul client este deblocat și conexiunea este stabilită. Atât clientul cât și serverul pot utiliza acum primitivele SEND și RECEIVE pentru a transmite sau recepționa date folosind o conexiune duplex integral. Se pot folosi și apelurile de sistem READ și WRITE standard din UNIX, dacă nu sunt necesare opțiunile speciale oferite de SEND și RECV.

Eliberarea conexiunii este simetrică. Atunci când ambele părți au executat primitiva CLOSE, conexiunea este eliberată.

6.1.4 Un exemplu de programare cu socluri: server de fișiere pentru Internet

Ca exemplu de cum pot fi folosite apelurile pentru socluri, vom considera codurile client și server din fig. 6-6. Aici avem un server de Internet foarte primitiv împreună cu un exemplu de client care îl utilizează. Codul are multe limitări (discutate mai jos), dar în principiu codul server poate fi compilat și rulat pe orice sistem UNIX conectat la Internet. Codul client poate fi apoi compilat și rulat pe orie altă mașină UNIX din Internet, oriunde în lume. Codul client poate fi executat cu parametrii adecvați pentru a obține orice fișier la care serverul are acces pe mașina sa. Fișierul este scris la ieșirea standard, care, desigur, poate fi redirectată spre un fișier sau spre o conductă (pipe).

Să ne uităm mai întâi la codul server. Acesta începe incluzând niște „header”-e (antete) standard între care ultimele 3 conțin principalele definiții și structuri de date care se referă la Internet. Apoi urmează o definire a *SERVER_PORT* (portului de server) ca 12345. Acest număr a fost ales arbitrar. Orice număr între 1024 și 65535 va funcționa la fel de bine atâta timp cât nu este utilizat de alte procese. Bineînțeles, clientul și serverul trebuie sa folosească același port. Dacă serverul va deveni vreodată un succes de talie mondială (improbabil, știind cât de primitiv este) îi va fi asignat un port permanent sub 1024 și va apărea la *www.iana.org*.

Următoarele două linii în codul server definesc două constante necesare. Prima determină dimensiunea zonei de memorie folosite pentru transferul de fișiere. A doua determină cât de multe conexiuni în așteptare pot fi reținute înainte ca cele care urmează să fie înlăturate după sosire.

După declarațiile variabilelor locale începe codul server. Acesta pornește cu inițializarea unei structuri de date care va ține adresa IP a serverului. Această structură de date va fi în curând asociată cu soclul serverului. Apelul către *memset* setează structura de date la 0. Cele trei atribuiri care îi urmează completează trei din câmpurile sale. Ultima dintre ele conține portul serverului. Funcțiile *htonl* și *htons* se referă la conversia valorilor într-un format standard astfel încât codul să ruleze corect atât pe mașini „big-endian” (de exemplu SPARC) cât și mașini „little-endian” (de exemplu Pentium). Semantica lor exactă nu este relevantă aici.

```
/* Această pagină conține un program client care poate cere un fișier de la programul server
de pe pagina următoare. Serverul răspunde trimițând întregul fișier.
*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345       /* arbitrar, dar clientul și serverul trebuie să fie de acord */
#define BUF_SIZE 4096                                   /* dimensiunea blocului de transfer */

int main(int argc, char **argv)
{
 int c, s, bytes;
 char buf[BUF_SIZE];          /* zona tampon de memorie pentru fișierul ce este recepționat */
 struct hostent *h;

                                                        /* informații despre server */
 struct sockaddr_in channel;
                                                            /* păstrează adresa IP */

 if (argc != 3) fatal("Usage: client server-name file-name");
 h = gethostbyname(argv[1]);
                                                        /* caută adresa IP a gazdei */
 if (!h) fatal("gethostbyname failed");

 s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
 if (s < 0) fatal("socket");
 memset(&channel, 0, sizeof(channel));
 channel.sin_family= AF_INET;
 memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
 channel.sin_port= htons(SERVER_PORT);

 c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
 if (c < 0) fatal("connect failed");

 /* Conexiunea este acum stabilită. Trimite numele fișierului incluzând terminatorul de șir */
 write(s, argv[2], strlen(argv[2])+1);

 /* la fișierul și-l afișează la ieșirea standard. */
 while (1) {
    bytes = read(s, buf, BUF_SIZE);                          /* citește de la soclu */
    if (bytes <= 0) exit(0);                          /* verifică dacă este sfârșit de fișier*/
    write(1, buf, bytes);
                                                        /* scrie la ieșirea standard */
 }
}

fatal(char *string)
{
 printf("%s\n", string);
 exit(1);
}
```

Fig. 6-6. Codul client folosind socluri. Codul server este pe pagina următoare.

```
#include <sys/types.h>
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345       /* arbitrar, dar clientul și serverul trebuie să fie de acord */
#define BUF_SIZE 4096                                   /* dimensiunea blocului de transfer */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
 int s, b, l, fd, sa, bytes, on = 1;
 char buf[BUF_SIZE];           /* zona tampon de memorie pentru fișierul care este transmis */
 struct sockaddr_in channel;
                                                                  /* păstrează adresa IP */

 /* Construiește structura adresei pentru a se lega la soclu. */
 memset(&channel, 0, sizeof(channel));                                  /* canalul zero */
 channel.sin_family = AF_INET;
 channel.sin_addr.s_addr = htonl(INADDR_ANY);
 channel.sin_port = htons(SERVER_PORT);

 /* Deschidere pasivă. Așteaptă conexiunea. */
 s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);                        /* creează soclu */
 if (s < 0) fatal("socket failed");
 setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

 b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
 if (b < 0) fatal("bind failed");

 l = listen(s, QUEUE_SIZE);                                  /* specifică dimensiunea cozii */
 if (l < 0) fatal("listen failed");

 /* Soclul este acum setat și legat. Așteaptă conexiunea și o procesează. */
 while (1) {
    sa = accept(s, 0, 0);                              /* blocare pentru cererea de conexiune */
    if (sa < 0) fatal("accept failed");

    read(sa, buf, BUF_SIZE);                          /* citește numele fișierului de la soclu */

    /* Preia și returnează fișierul. */
    fd = open(buf, O_RDONLY);                           /* deschide fișierul de trimis înapoi */
    if (fd < 0) fatal("open failed");

    while (1) {
       bytes = read(fd, buf, BUF_SIZE); /* citește din fișier */
       if (bytes <= 0) break;                           /* verifică dacă este sfârșit de fișier */
       write(sa, buf, bytes);                                    /* scrie octeți la soclu */
     }
     close(fd);                                                        * închide fișierul */
    close(sa);                                                    /* închide conexiunea*/
 }
}
```

Fig. 6-6. Codul server. Codul client este pe pagina anterioară.

În continuare serverul creează un soclu și face verificare pentru erori (indicate de $s<0$). Într-o versiune de producție a codului mesajul de eroare ar putea fi mai explicit. Apelul către *setsockopt* este necesar pentru a permite portului să fie folosit de serverul care rulează la nesfârșit, răspunzând la cerere după cerere. Acum, adresa IP este legată la soclu și este făcută o verificare pentru a vedea dacă apelul către *bind* a reușit. Ultimul pas în inițializare este apelul către *listen* pentru a anunța acordul serverului de a accepta apeluri și pentru a spune sistemului să mențină un număr de până la *QUEUE_SIZE* din acestea în caz că ajung noi cereri în timp ce serverul o procesează încă pe cea curentă. Dacă coada este plină și ajung cereri suplimentare, se renunță la acestea.

În acest punct, serverul intră în bucla sa principală, pe care nu o mai părăsește niciodată. Singura cale de a-l opri este de a-l termina forțat din afară. Apelul la *accept* blochează serverul până când un client încearcă să stabilească o conexiune cu el. Dacă apelul *accept* e făcut cu succes, returnează un descriptor de fișier care poate fi folosit pentru citire și scriere, în mod asemănător descriptorilor ce sunt folosiți pentru a citi și a scrie în pipe-uri (tuburi). Cu toate acestea, spre deosebire de tuburi, care sunt unidirecționale, soclurile sunt bidirecționale, astfel că *sa* (socket address – adresa soclului) poate fi folosită și pentru citire din conexiune, și pentru a scrie pe ea.

După ce conexiunea este stabilită, serverul citește numele fișierului din ea. Dacă numele nu este încă disponibil, serverul se blochează așteptându-l. După ce ia numele fișierului, serverul deschide fișierul și intră într-o buclă care citește alternativ blocuri din fișier și le scrie pe soclu până când întregul fișier a fost copiat. Apoi serverul închide iar fișierul și conexiunea și așteaptă să apară următoarea conexiune. El repetă această buclă la infinit.

Acum să privim codul client. Pentru a înțelege cum funcționează, este necesar să înțelegem cum este invocat. Presupunând că este numit *client*, un apel tipic este

```
client flits.cs.vu.nl /usr/tom/filename >f
```

Acest apel funcționează doar dacă serverul rulează deja la *flits.cs.vu.nl* și fișierul */usr/tom/filemane* există și serverul are drept de citire pentru el. Dacă apelul are succes, fișierul este transferat prin Internet și scris în *f* , după care programul client se termină. Din moment ce serverul continuă după un transfer, clientul poate fi pornit din nou pentru a lua alte fișiere.

Codul client începe cu câteva directive *include* și declarații. Execuția începe verificând dacă a fost apelat cu număr corect de argumente (*argc*=3 înseamnă numele programului plus două argumente). Observați că *argv[1]*conține numele serverului (de exemplu *flits.cs.vu.nl*) și este convertit la o adresă IP către *gethostbyname*. Această funcție folosește DNS pentru a căuta numele. Vom studia DNS în cap. 7.

În continuare este creat și inițializat un soclu. Apoi, clientul încearcă să stabilească o conexiune TCP cu serverul, folosind *connect*. Dacă serverul funcționează pe mașina menționată și atașat la *SERVER_PORT* și este fie inactiv, fie are loc în coada sa *listen*, conexiunea va fi (în cele din urmă) stabilită. Folosind conexiunea, clientul trimite numele fișierului scriind pe soclu. Numărul de octeți trimiși este cu 1 mai mare decât numele, deoarece terminatorul de șir (un octet 0) trebuie de asemenea trimis pentru a spune serverului unde se sfârșește numele.

Acum clientul intră într-o buclă, citind fișierul bloc cu bloc de la soclu și copiindu-l la ieșirea standard. Când acestea se termină, pur și simplu iese.

Procedura *fatal* afișează un mesaj de eroare și iese. Serverul are nevoie de aceeași procedură, dar aceasta a fost omisă datorită lipsei de spațiu pe pagina. Din moment ce clientul și serverul sunt compilate separat și în mod normal rulează pe calculatoare diferite, ele nu pot partaja codul procedurii *fatal*.

Aceste două programe (la fel ca și orice material referitor la această carte) pot fi luate de la adresa de Web a cărții

http://www.prenhall.com/tanenbaum

dând clic pe link-ul către situl Web de lângă fotografia coperții. Ele pot fi descărcate și compilate pe orice sisteme UNIX (de exemplu Solaris, BSD, Linux) cu comenzile:

```
cc –o client client.c –lsocket –lnsl
cc –o server sever.c –lsocket –lnsl
```

Serverul este pornit tastând doar

```
server
```

Clientul are nevoie de două argumente, așa cum s-a discutat mai sus. O versiune de Windows este de asemenea disponibilă pe situl Web.

Ca o observație, acest server nu este ultimul cuvânt în domeniul programelor server. Verificarea erorilor este ineficientă și raportarea erorilor este mediocră. În mod clar serverul nu a auzit niciodată de securitate, și folosirea doar a apelurilor de sistem UNIX nu este ultimul cuvânt în independența de platformă. De asemenea face unele presupuneri care sunt tehnic ilegale, cum ar fi presupunerea că numele fișierului începe în zona de memorie tampon și este transmis automat. Din moment ce tratează toate cererile strict secvențial (deoarece are doar un singur fir de execuție) performanța este slabă. În ciuda acestor neajunsuri, este un server de fișiere Internet complet și funcțional. În exerciții, cititorul este invitat să le îmbunătățească. Pentru mai multe informații despre programare cu socluri, a se vedea (Stevens, 1997).

6.2 NOȚIUNI DE BAZĂ DESPRE PROTOCOALELE DE TRANSPORT

Serviciul transport este implementat prin intermediul unui **protocol de transport** folosit de cele două entități de transport. Câteva caracteristici sunt asemănătoare pentru protocoalele de transport și pentru cele de legătură de date studiate în detaliu în cap. 3. Amândouă trebuie să se ocupe, printre altele, de controlul erorilor, de secvențiere și de controlul fluxului.

Totuși, există diferențe semnificative între cele două protocoale. Aceste diferențe sunt datorate deosebirilor majore dintre mediile în care operează protocoalele, așa cum rezultă din fig. 6-7. La nivelul legăturii de date, cele două rutere comunică direct printr-un canal fizic, în timp ce la nivelul transport acest canal fizic este înlocuit de întreaga subrețea. Această deosebire are mai multe implicații importante pentru protocoale, așa cum vom vedea în acest capitol.

În cazul legăturii de date, pentru un ruter nu trebuie specificat cu care alt ruter vrea să comunice, deoarece fiecare linie specifică în mod unic o destinație. În schimb, în cazul nivelului transport este necesară adresarea explicită.

În plus, procesul stabilirii unei conexiuni prin cablul din fig. 6-7(a) este simplu: celălalt capăt este întotdeauna acolo (în afară de cazul în care nu a ‘căzut’) și în nici unul din cazuri nu sunt prea multe de făcut. Pentru nivelul transport însă, stabilirea inițială a conexiunii este mult mai complicată, așa cum vom vedea.

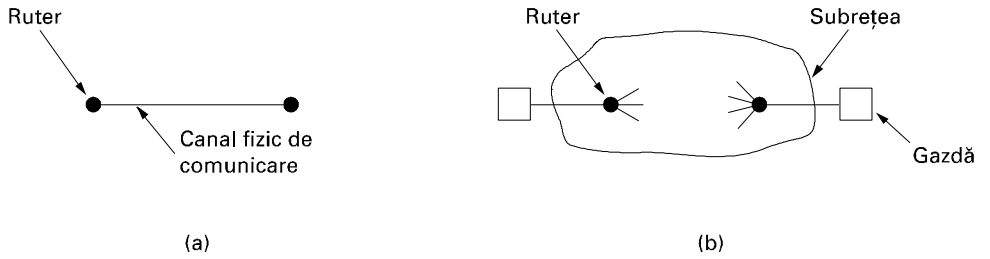


Fig. 6-7. (a) Mediul pentru nivelul legătură de date. (b) Mediul pentru nivelul transport.

O altă diferență între nivelurile legătură de date și transport, care generează multe probleme, este existența potențială a unei capacități de memorare a subrețelei. Atunci când un ruter trimite un cadru (nivel legătură de date), acesta poate să ajungă sau poate să se piardă, dar nu poate să se plimbe un timp ajungând până la capătul lumii și să apară 30 de secunde mai târziu, într-un moment nepotrivit. Dacă subrețeaua folosește datagrame și dirijare adaptivă, există o posibilitate - care nu poate fi neglijată - ca un pachet să fie păstrat pentru un număr oarecare de secunde și livrat mai târziu. Consecințele capacității de memorare a subrețelei pot fi uneori dezastruoase și necesită folosirea unor protocoale speciale.

O ultimă diferență între nivelurile legătură de date și transport este una de dimensionare și nu de proiectare. Folosirea tampoanelor și controlul fluxului sunt necesare la amândouă nivelurile, dar prezența unui număr mare de conexiuni în cazul nivelului transport necesită o abordare diferită de cea de la nivelul legătură de date. În cap. 3, unele protocoale alocau un număr fix de tampoane pentru fiecare linie, astfel încât atunci când sosea un cadru, exista întotdeauna un tampon disponibil. La nivel transport, numărul mare de conexiuni care trebuie să fie gestionate face ca ideea de a aloca tampoane dedicate să fie mai puțin atractivă. În următoarele secțiuni, vom examina atât aceste probleme importante cât și altele.

6.2.1 Adresarea

Atunci când un proces aplicație (de exemplu, un proces utilizator) dorește să stabilească o conexiune cu un proces aflat la distanță, el trebuie să specifice cu care proces dorește să se conecteze. (La protocoalele de transport neorientate pe conexiune apare aceeași problemă: cui trebuie trimis mesajul?). Metoda folosită în mod normal este de a defini adrese de transport la care procesele pot să aștepte cereri de conexiune. În Internet acestea se numesc porturi. La rețelele ATM perechile se numesc **AAL - SAP**-uri. În continuare vom folosi pentru acestea termenul generic **TSAP (Transport Service Access Point -** punct de acces la serviciul de transport). Punctele similare în cazul nivelului rețea (adică adresele la nivel rețea) sunt numite **NSAP (Network Service Access Point).** Adresele IP sunt exemple de NSAP-uri.

Fig. 6-8 ilustrează relația între TSAP, NSAP și conexiunile transport. Procesele aplicație, atât clienții cât și serverele, se pot atașa la TSAP pentru a stabili o conexiune la un TSAP la distanță. Aceste conexiuni rulează prin TSAP-uri pe fiecare gazdă așa cum se arată. Necesitatea de a avea mai multe TSAP-uri este dată de faptul că în unele rețele, fiecare calculator are un singur NSAP, deci cumva este nevoie să se distingă mai multe puncte de sfârșit de transport care partajează acel NSAP.

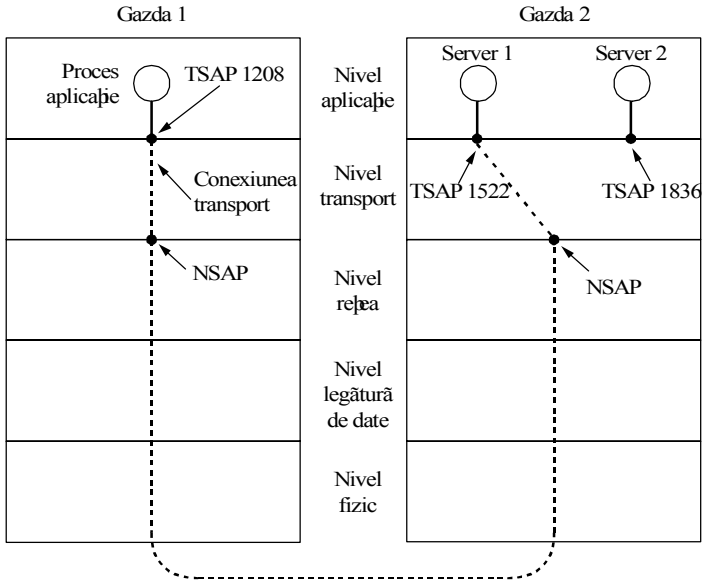


Fig. 6-8. TSAP, NSAP și conexiunile la nivel transport.

Un scenariu posibil pentru stabilirea unei conexiuni la nivel transport este următorul.

1. Un proces server care furnizează ora exactă și care rulează pe gazda 2 se atașează la TSAP 122 așteptând un apel. Felul în care un proces se atașează la un TSAP nu face parte din modelul de rețea și depinde numai de sistemul de operare local. Poate fi utilizat un apel de tip LISTEN din capitolul precedent.
2. Un proces aplicație de pe gazda 1 dorește să afle ora exactă; atunci el generează un apel CONNECT specificând TSAP 1208 ca sursă și TSAP 1522 ca destinație. Această acțiune are ca rezultat în cele din urmă stabilirea unei conexiuni la nivel transport între procesele aplicație de pe gazda 1 și serverul 1 de pe gazda 2.
3. Procesul aplicație trimite o cerere o cerere pentru timp.
4. Procesul server de timp răspunde cu timpul curent.
5. Conexiunea transport este apoi eliberată.

Trebuie reținut că foarte bine pot exista alte servere pe gazda 2 care să fie atașate la alte TSAP-uri și care să aștepte conexiuni care ajung pe același NSAP.

Fig. 6-8 explică aproape tot, cu excepția unei mici probleme: cum știe procesul utilizator de pe mașina 1 că serverul de oră exactă este atașat la TSAP 1522? O posibilitate este ca acest server de oră exactă să se atașeze la TSAP 1522 de ani de zile și, cu timpul, toți utilizatorii au aflat acest lucru. În acest model serviciile au adrese TSAP fixe, care pot fi afișate în fișiere în locuri bine cunoscute, cum este fișierul *etc/services* pe sistemele UNIX care afișează ce servere sunt atașate permanent și la ce porturi.

Dar schema cu adrese de servicii fixe funcționează doar pentru un număr mic de servicii cheie, a căror adresă nu se schimbă niciodată (de exemplu server de Web). Însă, în general, procesele utilizator vor să comunice cu alte procese care există numai pentru scurt timp și nu au o adresă TSAP dinainte cunoscută. Pe de altă parte, pot exista mai multe procese server, majoritatea utilizate foarte

rar, și ar fi neeconomic ca fiecare să fie activ și să asculte la o adresă TSAP fixă tot timpul. Pe scurt, este necesară o soluție mai bună.

O astfel de soluție este prezentată în fig. 6-9, într-o formă simplificată. Ea este cunoscută ca **protocolul de conectare inițială.** În loc ca orice server să asculte la un TSAP fixat, fiecare mașină care dorește să ofere servicii utilizatorilor aflați la distanță are un **server de procese (process server)** special care acționează ca un intermediar pentru toate serverele mai puțin utilizate. El ascultă în același timp la un număr de porturi, așteptând o cerere de conexiune. Utilizatorii potențiali ai serviciului încep prin a face o cerere de conexiune, specificând adresa TSAP a serviciului pe care îl doresc. Dacă nu există un server care să aștepte conexiuni la acel port, ele obțin o conexiune la serverul de procese, ca în fig. 6-9 (a).

După ce primește cererea, serverul de procese dă naștere serverului cerut, permițându-i să moștenească conexiunea cu procesul utilizator. Noul server execută prelucrarea cerută, în timp ce serverul de procese continuă să aștepte noi cereri, ca în fig. 6-9 (b).

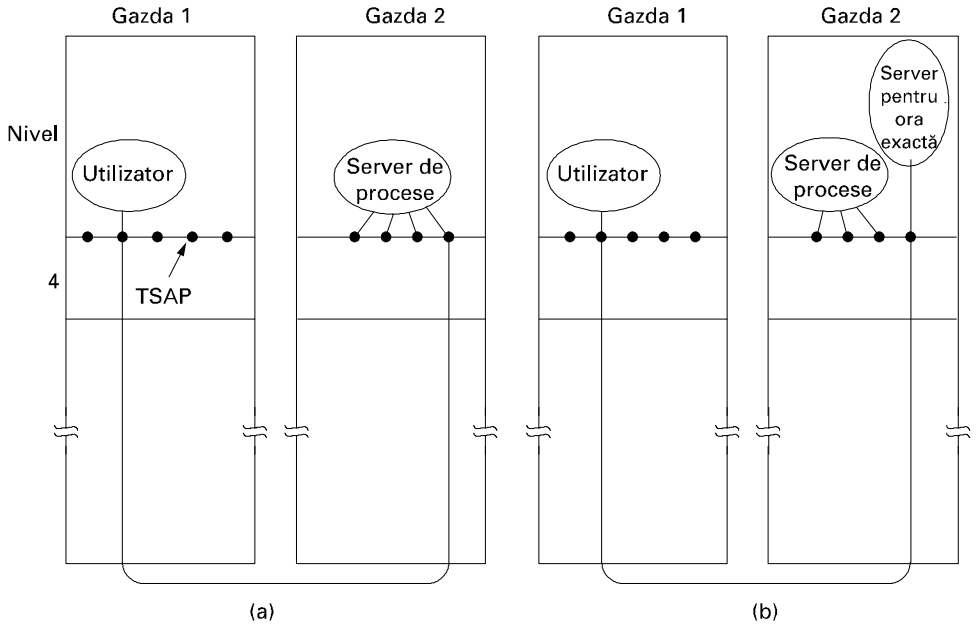


Fig. 6-9. Stabilirea unei conexiuni între calculatorul gazdă 1 și serverul pentru ora exactă.

În timp ce acest protocol funcționează bine pentru serverele care pot fi create ori de câte ori este nevoie de ele, există mai multe situații în care serviciile există independent de serverul de procese. De exemplu, un server de fișiere va rula folosind un hardware specializat (o mașina cu disc) și nu poate fi creat din mers.

Pentru a trata această situație, este des utilizată o soluție alternativă. În acest model există un proces special numit **server de nume** (**name server** sau, uneori, **directory server**). Pentru a găsi adresa TSAP corespunzătoare unui serviciu dat prin nume, așa cum este „ora exactă", utilizatorul stabilește o conexiune cu serverul de nume (care așteaptă mesaje la un TSAP cunoscut). Apoi utilizatorul trimite un mesaj specificând numele serviciului, iar serverul de nume îi trimite înapoi adresa TSAP a

acestuia. După aceasta, utilizatorul eliberează conexiunea cu serverul de nume și stabilește o nouă conexiune cu serviciul dorit.

În acest model, atunci când este creat un nou serviciu, el trebuie să se înregistreze singur la serverul de nume, furnizând atât numele serviciului oferit (în general un șir ASCII) cât și adresa TSAP. Serverul de nume înregistrează această informație într-o bază de date internă, astfel încât el va ști răspunsul atunci când vor sosi noi cereri.

Funcționarea serverului de nume este asemănătoare cu serviciul de informații de la un sistem telefonic: este furnizată corespondența dintre nume și numere de telefon. Ca și în cazul telefoanelor, este esențial ca adresa bine cunoscută a serverului de nume (sau a serverului de procese, în protocolul de conectare inițială) să fie într-adevăr bine cunoscută. Dacă nu știi numărul de la informații, nu poți afla nici un alt număr de telefon. Dacă crezi că numărul de la informații este evident pentru toți, încearcă să-l folosești și în altă țară!

6.2.2 Stabilirea conexiunii

Stabilirea unei conexiuni poate să pară ușoară dar, în realitate, este surprinzător de complicată. La prima vedere, ar părea suficient ca o entitate de transport să trimită numai un TPDU CONNECTION REQUEST și să aștepte replica CONNECTION ACCEPTED . Problema apare deoarece rețeaua poate pierde, memora sau duplica pachete. Acest comportament duce la complicații serioase.

Putem imagina o subrețea care este atât de congestionată încât confirmările ajung greu înapoi, și, din această cauză, fiecare pachet ajunge să fie retransmis de câteva ori. Putem presupune că subrețeaua folosește datagrame și fiecare pachet urmează un traseu diferit. Unele pachete pot să întâlnească o congestie locală de trafic și să întârzie foarte mult, ca și cum ar fi fost memorate de subrețea un timp și eliberate mai târziu.

Cel mai neplăcut scenariu ar fi: un utilizator stabilește o conexiune cu o bancă și trimite un mesaj cerând transferul unei sume de bani în contul unei alte persoane în care nu poate avea încredere în totalitate, și apoi eliberează conexiunea. Din nefericire, fiecare pachet din acest scenariu este duplicat și memorat în subrețea. După ce conexiunea a fost eliberată, pachetele memorate ies din subrețea și ajung la destinatar, cerând băncii să stabilească o nouă conexiune, să facă transferul (încă o dată) și să elibereze conexiunea. Banca nu poate să știe că acestea sunt duplicate, ea trebuie să presupună că este o tranzacție independentă și va transfera banii încă o dată. În continuarea acestei secțiuni, vom studia problema duplicatelor întârziate, punând accentul în mod special pe algoritmii pentru stabilirea sigură a conexiunilor, astfel încât scenarii ca cel de mai sus să nu poată să apară.

După cum am mai spus, punctul crucial al problemei este existența duplicatelor întârziate. El poate fi tratat în mai multe feluri, dar nici unul nu este într-adevăr satisfăcător. O posibilitate este de a utiliza adrese de transport valabile doar pentru o singură utilizare. În această abordare, ori de câte ori este necesară o adresă la nivel transport, va fi generată una nouă. După ce conexiunea este eliberată, adresa nu mai este folosită. Acest mecanism face însă imposibl modelul cu server de procese din fig. 6-9.

O altă posibilitate este de a atribui fiecărei conexiuni un identificator (adică, un număr de secvență incrementat pentru fiecare conexiune stabilită), ales de cel care inițiază conexiunea, și pus în fiecare TPDU, inclusiv în cel care inițiază conexiunea. După ce o conexiune este eliberată, fiecare entitate de transport va completa o tabelă cu conexiunile care nu mai sunt valide, reprezentate ca perechi (entitate de transport, identificator conexiune). Ori de câte ori apare o cerere de conexiune se va verifica în tabelă că ea nu aparține unei conexiuni care a fost eliberată anterior.

Din nefericire, această schemă are un defect important: ea necesită ca fiecare entitate de transport să mențină informația despre conexiunile precedente un timp nedefinit. Dacă o mașină cade și își pierde datele din memorie, ea nu va mai ști care identificatori de conexiune au fost deja utilizați.

Putem încerca și o altă soluție. În loc să permitem pachetelor să trăiască la nesfârșit în subrețea, putem inventa un mecanism care să elimine pachetele îmbătrânite. Dacă suntem siguri că nici un pachet nu poate să supraviețuiască mai mult de un anume interval de timp cunoscut, problema devine ceva mai ușor de rezolvat.

Durata de viață a pachetelor poate fi limitată la un maxim cunoscut, folosind una (sau mai multe) din următoarele tehnici:

1. Restricții în proiectarea subrețelei
2. Adăugarea unui contor al nodurilor parcurse în fiecare pachet
3. Adăugarea unei amprente de timp la fiecare pachet

Prima metodă include soluțiile care împiedică pachetele să stea în buclă, combinate cu modalități de a limita întârzierile datorate congestiilor, pe orice cale din rețea (indiferent de lungime). A doua metodă constă în a inițializa contorul cu o valoare adecvată și în a-l decrementa la trecerea prin orice nod. Protocolul de nivel rețea pur și simplu elimină pachetele al căror contor a devenit zero. A treia metodă presupune ca fiecare pachet să conțină timpul creării sale, ruterele acceptând să elimine pachetele mai vechi de un anumit moment de timp, asupra căruia au căzut de acord. Această metodă necesită ca ceasurile de la fiecare ruter să fie sincronizate, și această cerință în sine este destul de greu de îndeplinit (mai ușor este dacă sincronizarea ceasurilor se obține din exteriorul rețelei, de exemplu folosind GPS sau stații radio care transmit periodic ora exactă).

În practică, nu este suficient doar să garantăm că pachetul este eliminat, ci trebuie garantat și că toate confirmările sale au fost eliminate, astfel încât vom introduce T, care va fi un multiplu (mic) al duratei maxime de viață a unui pachet. Depinde de protocol de câte ori T este mai mare decât durata de viață a unui pachet. Dacă așteptăm un timp T după trimiterea unui pachet putem fi siguri că toate urmele sale au dispărut și nici el, nici vreo confirmare de-a sa nu vor apărea din senin, doar ca să complice lucrurile.

Folosind durata de viață limitată a pachetelor, există metode de a obține conexiuni sigure a căror corectitudine a fost demonstrată. Metoda descrisă în cele ce urmează este datorată lui Tomlinson (1975). Ea rezolvă problema, dar introduce câteva particularități proprii. Metoda a fost îmbunătățită de Sunshine și Dalal (1978). Variante ale sale sunt larg folosite în practică, inclusiv în TCP.

Pentru a ocoli problemele generate de pierderea tuturor datelor din memoria unei mașini după o cădere, Tomlinson propune echiparea fiecărei mașini cu un ceas. Nu este nevoie ca ceasurile de pe mașini diferite să fie sincronizate. Fiecare ceas va fi de fapt un contor binar care se autoincrementează după un anumit interval de timp. În plus, numărul de biți ai contorului trebuie să fie cel puțin egal cu numărul de biți al numerelor de secvență. În cele din urmă, și cel mai important, ceasul trebuie să continue să funcționeze chiar în cazul în care calculatorul gazdă cade.

Ideea de bază este de a fi siguri că două TPDU numerotate identic nu pot fi generate în același timp. Atunci când conexiunea este inițiată, k biți mai puțin semnificativi ai ceasului sunt folosiți ca număr inițial de secvență (tot k biți). Astfel, fiecare conexiune începe să-și numeroteze TPDU-urile sale cu un număr de secvență diferit. Spațiul numerelor de secvență ar trebui să fie suficient de mare pentru ca, în timpul scurs până când contorul ajunge din nou la același număr, toate TPDU-urile vechi cu acel număr să fi dispărut deja. Această relație liniară între timp și numărul de secvență inițial este prezentată în fig. 6-10.

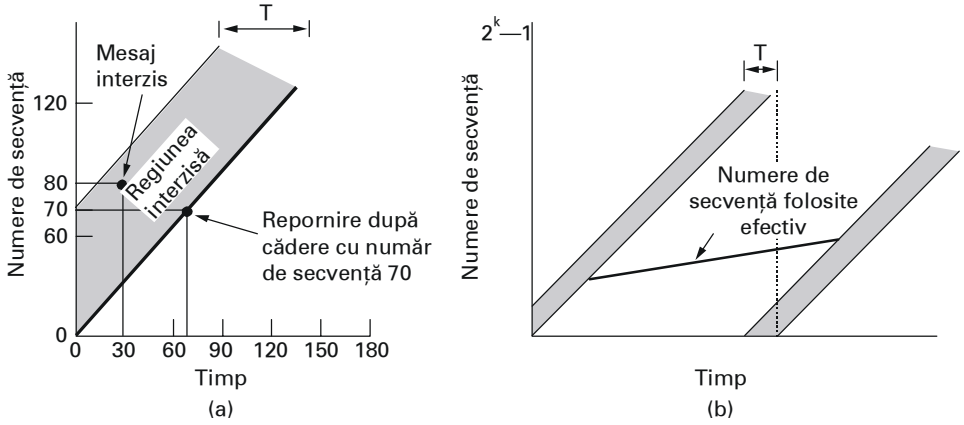


Fig. 6-10. (a) TPDU-urile nu pot să intre în regiunea interzisă. (b) Problema resincronizării.

Odată ce ambele entități de transport au căzut de acord asupra numărului de secvență inițial, pentru controlul fluxului poate fi folosit orice protocol cu fereastră glisantă. În realitate curba ce reprezintă numărul inițial de secvență (desenată cu linie îngroșată) nu este chiar liniară, ci în trepte, căci ceasul avansează în trepte. Pentru simplitate, vom ignora acest detaliu.

O problemă apare atunci când cade un calculator gazdă. Când el își revine, entitatea sa de transport nu știe unde a rămas în spațiul numerelor de secvență. O soluție este de a cere entității de transport să stea neocupată T secunde după revenire pentru ca în acest timp toate vechile TPDU să dispară. Totuși, într-o rețea complexă T poate fi destul de mare, astfel că această strategie nu este prea atrăgătoare.

Pentru a evita cele T secunde de timp nefolosit după o cădere, este necesar să introducem o nouă restricție în utilizarea numerelor de secvență. Necesitatea introducerii acestei restricții este evidentă în următorul exemplu. Fie T, timpul maxim de viață al unui pachet, egal cu 60 de secunde și să presupunem că ceasul este incrementat la fiecare secundă. După cum arată linia îngroșată din fig. 6-10(a), numărul inițial de secvență pentru o conexiune inițiată la momentul x este x. Să ne imaginăm că la t=30 sec, unui TPDU trimis pe conexiunea cu numărul 5 (deschisă anterior) i se dă numărul de secvență 80. Să numim acest TPDU *X.* Imediat după ce *X* este trimis, calculatorul gazdă cade și revine imediat. La t=60 el redeschide conexiunile de la 0 la 4. La t=70, el deschide conexiunea 5, folosind un număr de secvență inițial 70, așa cum am stabilit. În următoarele 15 secunde el va transmite TPDU-uri cu date numerotate de la 70 la 80. Astfel că la t=85, în subrețea este generat un nou TPDU cu numărul de secvență 80 și conexiunea 5. Din nefericire, TPDU *X* încă mai există. Dacă el ajunge înaintea noului TPDU 80, atunci TPDU *X* va fi acceptat și TPDU-ul corect va fi respins ca fiind un duplicat.

Pentru a preveni o astfel de problemă trebuie să luăm măsuri ca numerele de secvență să nu fie utilizate (adică atribuite unor noi TPDU-uri) un timp T înaintea utilizării lor ca noi numere de secvență. Combinațiile imposibile - timp, număr de secvență - sunt prezentate în fig. 6-10(a) ca **regiunea interzisă**. Înainte de trimiterea oricărui TPDU pe orice conexiune, entitatea de transport trebuie să citească ceasul și să verifice dacă nu cumva se află în regiunea interzisă.

Pot să apară probleme în două cazuri: dacă un calculator gazdă trimite prea multe date și prea repede pe o conexiune nou deschisă, curba numărului de secvență în funcție de timp poate să fie mult

mai abruptă decât cea inițială. Aceasta înseamnă că rata de transmisie pentru orice conexiune este de cel mult un TPDU pe unitatea de timp a ceasului. De asemenea, este necesar ca entitatea de transport să aștepte până când ceasul avansează o dată, înainte să deschidă o nouă conexiune pentru ca, la revenirea după o cădere, același număr de secvență să nu fie utilizat de două ori. Cele două observații de mai sus sunt argumente pentru ca perioada ceasului să fie cât mai mică (câteva μs sau mai mică).

Din nefericire, intrarea în regiunea interzisă prin trimitere prea rapidă nu este singura situație care creează probleme. Fig. 6-10(b) arată că la orice rată de transfer mai mică decât frecvența ceasului curba numerelor de secvență utilizate raportată la timp va ajunge până la urmă în regiunea interzisă din stânga. Cu cât curba numerelor de secvență utilizate va fi mai înclinată, cu atât mai târziu se ajunge în regiunea interzisă. Așa cum am afirmat anterior, imediat înaintea trimiterii unui TPDU, entitatea de transport trebuie să verifice dacă nu se află cumva în regiunea interzisă, și, dacă se află, să întârzie transmisia cu T secunde sau să resincronizeze numerele de secvență.

Metoda bazată pe ceasuri rezolvă problema duplicatelor întârziate pentru TPDU-urile de date, dar pentru ca această metodă să poată fi folosită, trebuie mai întâi să stabilim conexiunea. Deoarece TPDU-urile de control pot și ele să fie întârziate, pot apărea probleme atunci când entitățile de transport încercă să cadă de acord asupra numărului inițial de secvență. Să presupunem, de exemplu, că, pentru a stabili o conexiune, gazda 1 trimite un mesaj CONNECTION REQUEST conținând numărul de secvență inițial propus și portul destinație gazdei 2. Acesta va confirma mesajul trimițând înapoi un TPDU CONNECTION ACCEPTED . Dacă TPDU-ul CONNECTION REQUEST este pierdut, dar un duplicat întârziat al unui alt CONNECTION REQUEST va ajunge la gazda 2, atunci conexiunea nu va fi stabilită corect.

Pentru a rezolva aceasta problemă, Tomlinson (1975) a introdus stabilirea conexiunii cu înțelegere în trei pași (three-way handshake). Acest protocol nu necesită ca ambele părți să înceapă să trimită același număr de secvență, deci poate fi utilizat și împreună cu alte metode de sincronizare decât ceasul global. Procedura normală de inițiere a conexiunii este exemplificată în fig. 6-11(a). Gazda 1 alege un număr de secvență *x* și trimite un TPDU CONNECTION REQUEST care conține *x* gazdei 2. Gazda 2 răspunde cu CONNECTION ACK, confirmând *x* și anunțând numărul său inițial de secvență, *y*. În cele din urmă gazda 1 confirmă alegerea lui *y* gazdei 2 în primul mesaj de date pe care îl trimite.

Vom arunca acum o privire asupra stabilirii conexiunii cu înțelegere în trei pași în prezența TPDU-urilor de control duplicate întârziate. În fig. 6-11(b) primul TPDU sosit este o copie întârziată a unui CONNECTION REQUEST de la o conexiune mai veche. Acest TDU ajunge la gazda 2 fără ca gazda 1 să știe. Gazda 2 răspunde acestui TPDU trimițând gazdei 1 un TPDU ACK, verificând de fapt că gazda 1 a încercat într-adevăr să stabilească o conexiune. Atunci când gazda 1 refuză cererea gazdei 2 de a stabili conexiunea, gazda 2 își dă seama că a fost păcălită de o copie întârziată și abandonează conexiunea. În acest fel o copie întârziată nu poate să strice nimic.

În cel mai rău caz, atât CONNECTION REQUEST cât și ACK sunt copii întârziate în subrețea. Acest caz este prezentat în 6-11(c). Ca și în exemplul precedent, gazda 2 primește o comandă CONNECTION REQUEST întârziată și răspunde la ea. În acest moment este extrem de important să ne aducem aminte că gazda 2 a propus *y* ca număr inițial de secvență pentru traficul de la 2 la 1, fiind sigur că nu mai există în rețea nici un TPDU (sau confirmare) cu același număr de secvență. Atunci când al doilea TPDU întârziat ajunge la gazda 2, aceasta deduce, din faptul că a fost confirmat *z* și nu *y,* că are de-a face cu o copie mai veche. Important este că nu există nici o combinație posibilă ale unor copii vechi ale TPDU-urilor întârziate care să reușească să inițieze o conexiune atunci când nimeni nu a cerut asta.

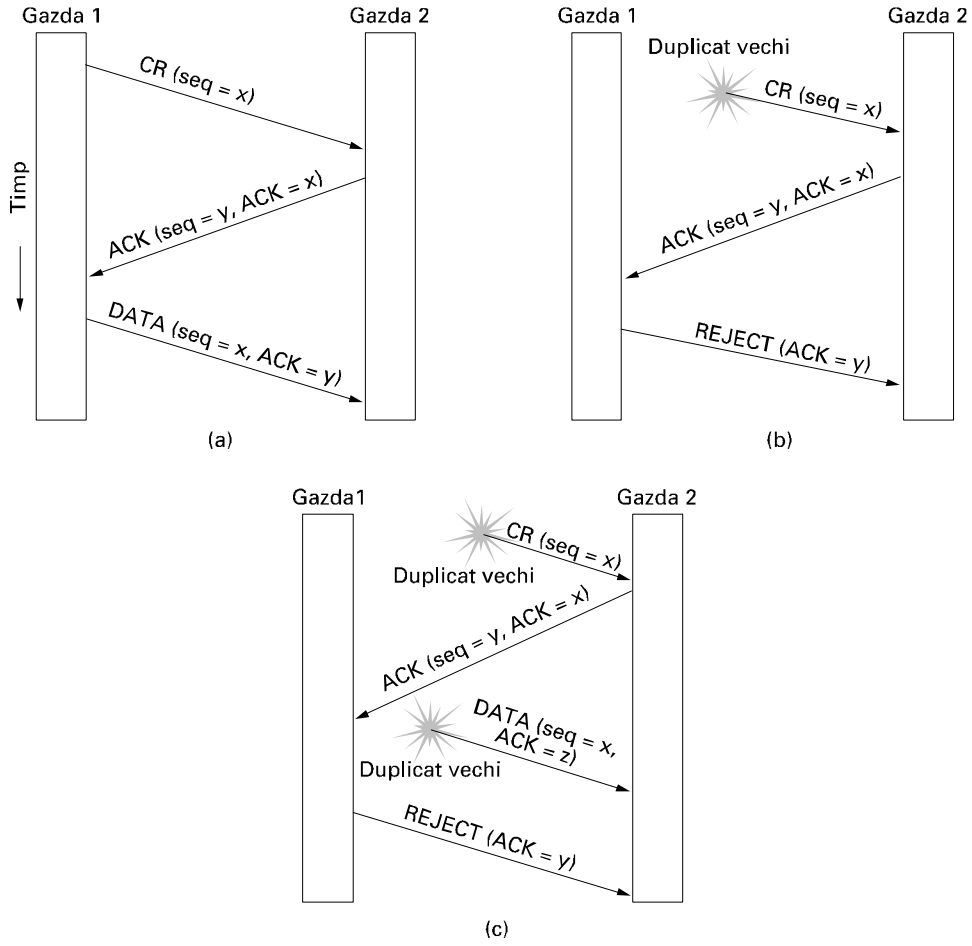


Fig. 6-11. Trei scenarii posibile de stabilire a conexiunii pentru un protocol cu înțelegere în trei pași. CR reprezintă CONNECTION REQUEST. (a) Cazul normal, (b) Un duplicat vechi al unui mesaj CONNECTION REQUEST apare când nu trebuie, (c) Sunt duplicate atât CONNECTION REQUEST cât și CONNECTION ACCEPTED.

6.2.3 Eliberarea conexiunii

Eliberarea unei conexiuni este mai ușoară decât stabilirea ei. Totuși, există mai multe dificultăți decât ne-am aștepta. Așa cum am mai amintit, există două moduri de a termina o conexiune: eliberare simetrică și eliberare asimetrică. Sistemul telefonic folosește eliberarea asimetrică: atunci când unul din interlocutori închide, conexiunea este întreruptă. Eliberarea simetrică privește conexiunea ca pe două conexiuni separate unidirecționale și cere ca fiecare să fie eliberată separat.

Eliberarea asimetrică este bruscă și poate genera pierderi de date. Să considerăm scenariul din fig. 6-12. După stabilirea conexiunii, gazda 1 trimite un TPDU care ajunge corect la gazda 2. Gazda

1 mai trimite un TPDU dar, înainte ca acesta să ajungă la destinație, gazda 2 trimite DISCONNECT REQUEST . În acest caz, conexiunea va fi eliberată și vor fi pierdute date.

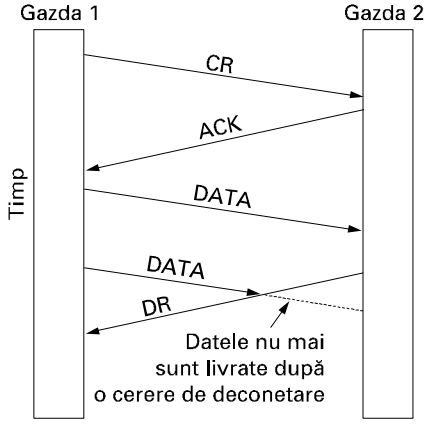


Fig. 6-12. Deconectare bruscă cu pierdere de date. CR= CONNECTION REQUEST, ACK=CONNECTION ACCEPTED , DR=DISCONNECT REQUEST.

Evident, pentru a evita pierderea de date, este necesar un protocol de eliberare a conexiunii mai sofisticat. O posibilitate este utilizarea eliberării simetrice: fiecare direcție este eliberată independent de cealaltă; un calculator gazdă poate să continue să primească date chiar și după ce a trimis un TPDU de eliberare a conexiunii.

Eliberarea simetrică este utilă atunci când fiecare proces are o cantitate fixă de date de trimis și știe bine când trebuie să transmită și când a terminat. În alte situații însă, nu este deloc ușor de determinat când trebuie eliberată conexiunea și când a fost trimis tot ce era de transmis. S-ar putea avea în vedere un protocol de tipul următor: atunci când 1 termină, trimite ceva de tipul: Am terminat. Ai terminat și tu? Dacă gazda 2 răspunde: Da, am terminat. Închidem! conexiunea poate fi eliberată în condiții bune.

Din nefericire, acest protocol nu merge întotdeauna. Binecunoscuta **problemă a celor două armate** este similară acestei situații: să ne imaginăm că armată albă și-a pus tabăra într-o vale (ca în fig. 6-13) Pe amândouă crestele care mărginesc valea sunt armatele albastre. Armata albă este mai mare decât fiecare din cele două armate albastre, dar împreună armatele albastre sunt mai puternice. Dacă oricare din armatele albastre atacă singură, ea va fi înfrântă, dar dacă ele atacă simultan, atunci vor fi victorioase.

Armatele albastre vor să-și sincronizeze atacul. Totuși singura lor posibilitate de comunicație este să trimită un mesager care să străbată valea. Mesagerul poate fi capturat de armata albă și mesajul poate fi pierdut (adică vor trebui să utilizeze un canal de comunicație nesigur). Problema este următoarea: există vreun protocol care să permită armatelor albastre să învingă?

Să presupunem că comandantul primei armate albastre trimite un mesaj: „Propun să atacăm pe 29 martie”, mesajul ajunge la armata 2 al cărei comandant răspunde: „De acord” iar răspunsul ajunge înapoi la armata 1. Va avea loc atacul în acest caz? Probabil că nu, deoarece comandantul armatei 2 nu știe dacă răspunsul său a ajuns sau nu la destinație. Dacă nu a ajuns, armata 1 nu va ataca, deci ar fi o prostie din partea lui să intre în luptă.

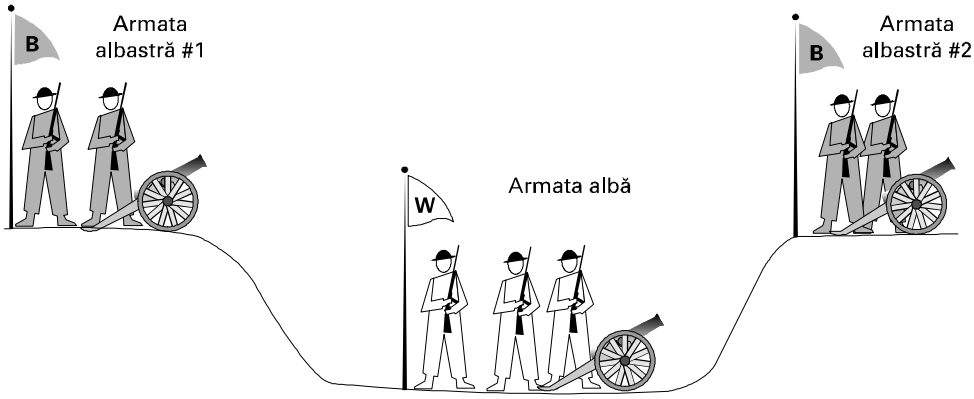


Fig. 6-13. Problema celor două armate.

Să încercăm să îmbunătățim protocolul, transformându-l într-unul cu înțelegere în trei pași. Inițiatorul propunerii de atac trebuie să confirme răspunsul. Presupunând că nici un mesaj nu este pierdut, armata 2 va avea confirmarea, dar comandantul armatei 1 va ezita acum. Până la urmă, el nu știe dacă confirmarea sa a ajuns la destinație și este sigur că dacă aceasta nu a ajuns, armata 2 nu va ataca. Am putea să încercăm un protocol cu confirmare în patru timpi, dar ne-am lovi de aceleași probleme.

De fapt, poate fi demonstrat că nu există un protocol care să funcționeze. Să presupunem că ar exista un asemenea protocol: decizia finală poate să depindă sau nu de ultimul mesaj al unui asemenea protocol. Dacă nu depinde, putem elimina acest mesaj (și oricare altul la fel) până ajungem la un protocol în care orice mesaj este vital. Ce se va întâmpla dacă ultimul mesaj este interceptat? Tocmai am hotărât că acest mesaj era unul vital, deci dacă este pierdut, atacul nu va avea loc. Deoarece cel care trimite ultimul mesaj nu poate fi niciodată sigur că mesajul a ajuns, el nu va risca atacând. Mai rău chiar, cealaltă armată albastră știe și ea acest lucru, deci nu va ataca nici ea.

Pentru a vedea legătura problemei celor două armate cu problema eliberării conexiunii este suficient să înlocuim 'atac' cu 'deconectare'. Dacă niciuna din părți nu se deconectează până nu este sigură că cealaltă parte este gata să se deconecteze la rândul ei, atunci deconectarea nu va mai avea loc niciodată.

În practică suntem dispuși să ne asumăm mai multe riscuri atunci când este vorba de eliberarea conexiunii decât atunci când este vorba de atacarea armatei albe, așa încât situația nu este întru totul fără speranță. Fig. 6-14 prezintă patru scenarii de eliberare a conexiunii folosind un protocol cu confirmare în trei timpi. Deși acest protocol nu este infailibil, el este în general adecvat.

În fig. 6-14(a) apare cazul normal în care unul dintre utilizatori trimite un TPDU de tip DR (DISCONNECT REQUEST) pentru a iniția eliberarea conexiunii. Atunci când acesta sosește, receptorul trimite înapoi tot un TPDU DR și pornește un ceas pentru a trata cazul în care mesajul său este pierdut. Când primește mesajul înapoi, inițiatorul trimite o confirmare și eliberează conexiunea. În sfârșit, la primirea confirmării, receptorul eliberează și el conexiunea. Eliberarea conexiunii înseamnă de fapt că entitatea de transport șterge din tabelele sale informația despre conexiunea respectivă din tabela de conexiuni deschise în momentul curent și semnalează acest lucru utilizatorului nivelului transport. Această acțiune nu este același lucru cu apelul unei primitive DISCONNECT de către un utilizator al nivelului transport.

Dacă ultima confirmare este pierdută, ca în fig. 6-14(b), putem salva situația cu ajutorul ceasului: după scurgerea unui anumit interval de timp conexiunea este eliberată oricum.

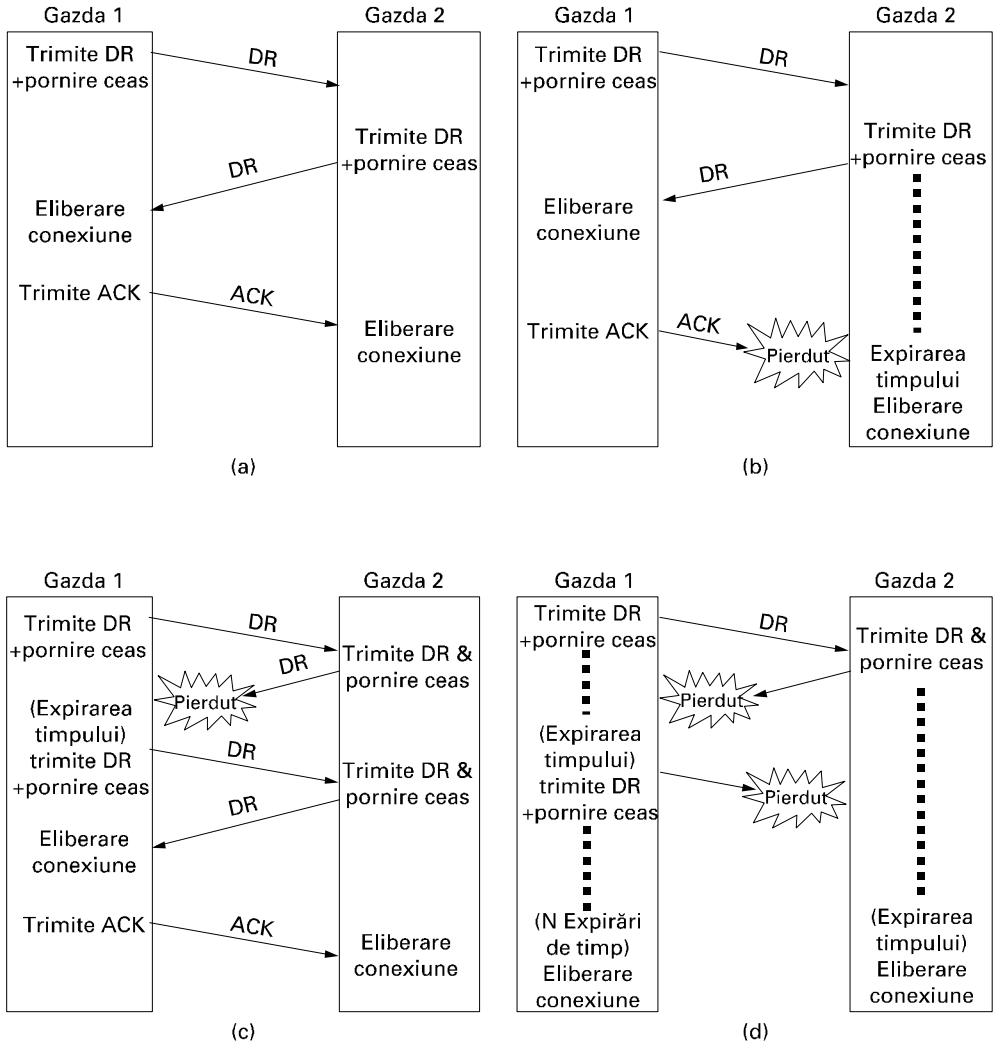


Fig. 6-14. Patru cazuri posibile la eliberarea conexiunii: (a)Cazul normal cu confirmare în trei timpi. (b)Ultima confirmare este pierdută. (c) Răspunsul este pierdut. (d) Răspunsul și următoarele cereri de deconectare sunt pierdute. (DR=DISCONNECT REQUEST).

Să considerăm acum cazul în care cel de-al doilea DR este pierdut: utilizatorul care a inițiat deconectarea nu va primi răspunsul așteptat, va aștepta un anumit timp și va trimite din nou un DR. În fig. 6-14(c), putem vedea cum se petrec lucrurile în acest caz, presupunând că la a doua încercare toate TPDU-urile ajung corect și la timp.

Ultima posibilitate pe care o studiem, prezentată în fig. 6-14(d), este similară cu cea din 6-14(c), cu următoarea diferență: de aceasta dată niciuna din încercările următoare de a retransmite DR nu reușește. După *N* încercări, emițătorul va elibera pur și simplu conexiunea. În același timp, și receptorul va elibera conexiunea după expirarea timpului.

Deși acest protocol este, în general, destul de bun, în teorie el poate să dea greș dacă atât mesajul DR inițial cât și *N* retransmisii ale sale se pierd. Emițătorul va renunța și va elibera conexiunea, în timp ce la celălalt capăt nu se știe nimic despre încercările de deconectare și aceasta va rămâne în continuare activă. În această situație rezultă o conexiune deschisă pe jumătate.

Am putea evita această problemă nepermițând emițătorului să cedeze după *N* reîncercări nereușite, ci cerându-i să continue până primește un răspuns. Totuși, dacă celeilalte părți i se permite să elibereze conexiunea după un interval de timp, este posibil ca inițiatorul să ajungă să aștepte la infinit. Dacă însă nu s-ar permite eliberarea conexiunii după expirarea unui interval de timp, atunci în cazul din fig. 6-14 (b) protocolul s-ar bloca.

O altă posibilitate de a scăpa de conexiunile pe jumătate deschise este de a aplica o regulă de tipul: dacă nici un TPDU nu sosește într-un anumit interval de timp, atunci conexiunea este eliberată automat. În acest fel, dacă una din părți se deconectează, cealaltă parte va detecta lipsa de activitate și se va deconecta și ea. Desigur, pentru a implementa aceasta regulă este nevoie ca fiecare entitate de transport să aibă un ceas care va fi repornit la trimiterea oricărui TPDU. La expirarea timpului, se transmite un TPDU vid, doar pentru a menține conexiunea deschisă. Pe de altă parte, dacă este aleasă această soluție, și câteva TPDU-uri vide sunt pierdute la rând pe o conexiune altfel liberă, este posibil ca, mai întâi una din părți, apoi cealaltă să se deconecteze automat.

Nu vom mai continua să detaliem acest subiect, dar probabil că acum este clar că eliberarea unei conexiuni fără pierderi de date nu este atât de simplă cum părea la început.

6.2.4 Controlul fluxului și memorarea temporară (buffering)

După ce am studiat în detaliu stabilirea și eliberarea conexiunii, vom arunca o privire asupra modului în care sunt tratate conexiunile cât timp sunt utilizate. Una din problemele cheie a apărut și până acum: controlul fluxului. La nivel transport există asemănări cu problema controlului fluxului la nivel legătură de date, dar există și deosebiri. Principala asemănare: la ambele niveluri este necesar un mecanism (fereastră glisantă sau altceva) pentru a împiedica un emițător prea rapid să depășească capacitatea de recepție a unui receptor prea lent. Principala deosebire: un ruter are în general puține linii, dar poate să aibă numeroase conexiuni. Această diferență face nepractică implementarea la nivel transport a strategiei de memorare temporară a mesajelor folosită la nivel legătură de date.

În protocoalele pentru legătura de date prezentate în cap. 3, cadrele sunt memorate temporar atât de ruterul care emite cât și de cel care recepționează. În protocolul 6, de exemplu, atât emițătorul cât și receptorul au alocate un număr de *MAXSEQ+1* tampoane pentru fiecare linie, jumătate pentru intrări și jumătate pentru ieșiri. Pentru un calculator gazdă cu, să spunem, 64 de conexiuni și numere de secvență de 4 biți, acest protocol ar necesita 1024 tampoane.

La nivel legătură de date, emițătorul trebuie să memoreze cadrele transmise, pentru că poate fi necesară retransmiterea acestora. Dacă subrețeaua oferă un serviciu datagramă, atunci entitatea de transport emițătoare va trebui să memoreze pachetele trimise din aceleași motive. Dacă receptorul știe că emițătorul stochează toate TPDU-urile până când acestea sunt confirmate, el poate să aloce sau nu tampoane specifice fiecărei conexiuni, după cum i se pare mai bine.

Receptorul poate, de exemplu, să rezerve un singur grup de tampoane pentru toate conexiunile. La sosirea unui TPDU se face o încercare de a obține dinamic un nou tampon. Dacă un tampon este liber, atunci TPDU-ul este acceptat, altfel, este refuzat. Cum emițătorul este gata să retransmită TPDU-urile pierdute de subrețea, faptul că unele TPDU-uri sunt refuzate nu produce nici o daună, deși în acest fel sunt risipite resurse. Emițătorul va retransmite până când va primi confirmarea.

Pe scurt, dacă serviciul rețea nu este sigur, emițătorul va trebui să memoreze toate TPDU-urile trimise, la fel ca la nivel legătură de date. Totuși, folosind un serviciu la nivel rețea sigur sunt posibile unele compromisuri. În particular, dacă emițătorul știe că receptorul are întotdeauna tampoane disponibile, atunci nu trebuie să păstreze copiile TPDU-urilor trimise. Totuși, dacă receptorul nu poate garanta că orice TPDU primit va fi acceptat, emițătorul va trebui să păstreze copii. În ultimul caz, emițătorul nu poate avea încredere în confirmarea primită la nivel rețea, deoarece aceasta confirmă sosirea TPDU-ului la destinație, dar nu și acceptarea lui. Vom reveni asupra acestui punct important mai târziu.

Chiar dacă receptorul va realiza memorarea temporară a mesajelor primite, mai rămâne problema dimensiunii tamponului. Dacă cea mai mare parte a TPDU-urilor au aceeași dimensiune, este naturală organizarea tampoanelor într-o resursă comună care conține tampoane de aceeași dimensiune, cu un TPDU per tampon, ca în fig. 6-15(a). Dacă însă dimensiunea TPDU-urilor variază de la câteva caractere tipărite la un terminal, la mii de caractere pentru un transfer de fișiere, organizarea ca o resursă comună cu tampoane de aceeași dimensiune va pune probleme. Dacă dimensiunea tampoanelor ar fi constantă, egală cu cel mai mare TPDU posibil, atunci va apărea o risipă de spațiu ori de câte ori este primit un TPDU mai scurt. Dacă dimensiunea tampoanelor este aleasă mai mică decât cel mai mare TPDU posibil, atunci pentru memorarea unui TPDU mai lung vor fi necesare mai multe tampoane, iar complexitatea operației va crește.

O altă soluție este utilizarea unor tampoane de dimensiune variabilă, ca în fig. 6-15(b). Avantajul este o mai bună utilizare a memoriei, cu prețul unei gestiuni a tampoanelor mai complicată. O a treia posibilitate este alocarea unui singur tampon circular pentru fiecare conexiune, ca în fig. 6-15(c). Această soluție are de asemenea avantajul unei utilizări eficiente a memoriei, dar numai în situația în care conexiunile sunt relativ încărcate.

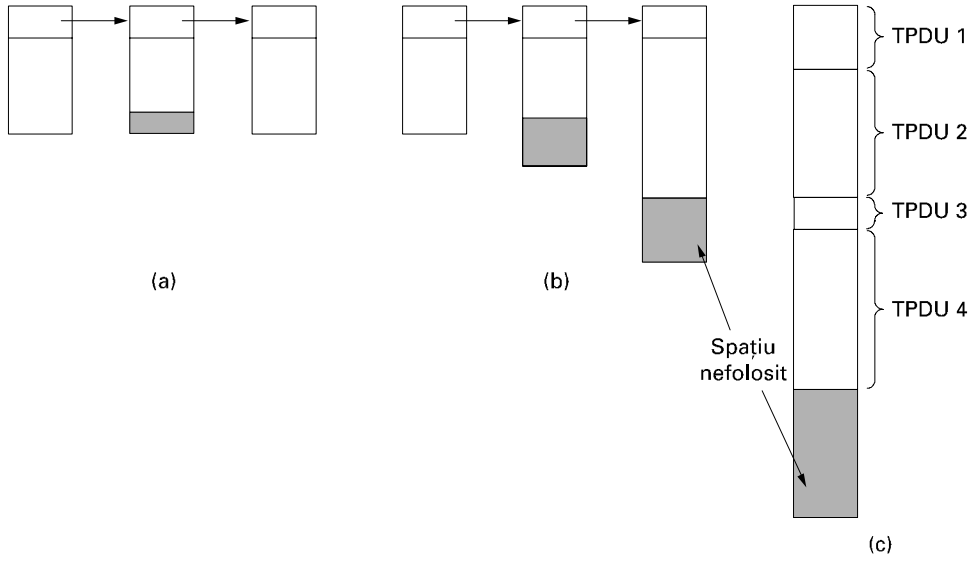


Fig. 6-15. (a) Tampoane de dimensiune fixă înlănțuite. (b) Tampoane de dimensiune variabilă înlănțuite. (c) Un tampon circular pentru fiecare conexiune.

Compromisul optim între memorarea temporară la sursă sau la destinație depinde de tipul traficului prin conexiune. Pentru un trafic în rafală cu o bandă de transfer îngustă, ca traficul produs de

un terminal interactiv, este mai bine ca tampoanele să nu fie prealocate, ci mai curând, alocate dinamic. Întrucât emițătorul nu poate să fie sigur că receptorul va reuși să aloce un tampon la sosirea unui pachet, emițătorul va fi nevoit să rețină copia unui TPDU transmis până când acesta va fi confirmat. Pe de altă parte, pentru un transfer de fișiere sau pentru orice alt trafic care necesită o bandă de transfer largă este mai bine dacă receptorul alocă un set întreg de tampoane, pentru a permite un flux de date la viteză maximă. Cu alte cuvinte, pentru un trafic în rafală cu o bandă de transfer îngustă este mai bine să fie folosite tampoane la emițător, în timp ce pentru un trafic continuu cu o bandă de transfer largă, este mai eficientă folosirea tampoanelor la receptor.

Pe măsură ce conexiunile sunt create și eliberate de trafic, iar șablonul se schimbă, emițătorul și receptorul trebuie să își ajusteze dinamic politica de alocare a tampoanelor. În consecință, protocolul de transport trebuie să permită emițătorului să ceară spațiu pentru tampoane la capătul celălalt al conexiunii. Tampoanele pot fi alocate pentru o anumită conexiune sau pot fi comune pentru toate conexiunile între două calculatoare gazdă. O alternativă este ca receptorul, cunoscând situația tampoanelor sale (dar necunoscând șablonul traficului) să poată spune emițătorului: „Am rezervat *X* tampoane pentru tine". Dacă numărul conexiunilor deschise trebuie să crească, poate fi necesar ca spațiul alocat unei singure conexiuni să scadă, deci protocolul trebuie să furnizeze și această facilitate.

O modalitate înțeleaptă de a trata alocarea dinamică a tampoanelor este separarea stocării în tampoane de confirmarea mesajelor, spre deosebire de protocolul cu fereastră glisantă din cap. 3. Alocarea dinamică a tampoanelor înseamnă, de fapt, o fereastră cu dimensiune variabilă. La început, emițătorul trimite cereri pentru un anumit număr de tampoane bazându-se pe o estimare a necesităților. Receptorul îi alocă atâtea tampoane cât își poate permite. De fiecare dată când emițătorul trimite un TPDU, el decrementează numărul de tampoane pe care le are alocate la receptor, oprindu-se când acest număr devine zero. Receptorul trimite înapoi confirmări și situația tampoanelor alocate, împreună cu traficul în sens invers.

Fig. 6-16 este un exemplu pentru modul în care administrarea dinamică a ferestrelor poate fi folosită într-o subrețea cu datagrame, cu numere de secvență pe 4 biți. Să presupunem că informația despre alocarea tampoanelor este împachetată în TPDU-uri distincte și că este separată de traficul în sens invers. La început A dorește opt tampoane, dar nu i se acordă decât patru. După aceea trimite trei TPDU-uri, iar al treilea este pierdut. TPDU-ul 6 confirmă recepția tuturor TPDU-urilor cu numere de secvență mai mici sau egale cu 1, permițând lui A să elibereze acele tampoane și, mai mult, îl informează pe A că B poate să mai recepționeze trei TPDU-uri (adică TPDU-urile cu numere de secvență 2, 3, 4). A știe că a trimis deja numărul 2, deci se gândește că ar putea trimite 3 și 4, ceea ce și încearcă să facă. În acest moment, el este blocat și trebuie să aștepte alocarea unui tampon. Expirarea timpului pentru primirea confirmării determină retransmisia mesajului 2 (linia 9), care poate avea loc, deși emițătorul este blocat, deoarece se vor utiliza tampoane deja alocate. În linia 10, B confirmă primirea tuturor TPDU-urilor până la 4, dar îl ține pe A încă blocat. O astfel de situație ar fi fost imposibilă cu protocolul cu fereastră glisantă prezentat în cap. 3. Următorul TPDU de la B la A alocă încă un tampon și îi permite lui A să continue.

În cazul strategiilor pentru alocarea tampoanelor de tipul celei de mai sus, eventuale probleme pot să apară în rețele neorientate pe conexiune dacă sunt pierdute TPDU-uri de control. Să privim linia 16 din fig. 6-16: B a mai alocat tampoane pentru A, dar TPDU-ul care transmitea această informație a fost pierdut. Deoarece TPDU-urile de control nu sunt numerotate sau retransmise, A va fi blocat. Pentru a preveni această situație, fiecare calculator gazdă trebuie să trimită periodic pe fiecare conexiune TPDU-uri de control ce pot conține confirmări și informații despre starea tampoanelor. În acest fel, A va fi deblocat mai devreme sau mai târziu.

	A	Mesajul	B	Comentarii
1	→	< cere 8 tampoane>	→	A cere 8 tampoane
2	←	<ack=15, buf=4>	←	B îi acordă tampoane numai de la 0 la 3
3	→	<seq = 0, data = m0>	→	A mai are 3 tampoane libere
4	→	<seq = 1, data = m1>	→	A mai are 2 tampoane libere
5	→	<seq = 2, data = m2>	•••	Mesaj pierdut, dar A crede că mai are un singur tampon liber
6	←	<ack = 1, buf = 3>	←	B confirmă 0 și 1 și permite 2-4
7	→	<seq = 3, data = m3>	→	A mai are tampoane
8	→	<seq = 4, data = m4>	→	A nu mai are tampoane libere și trebuie să se oprească
9	→	<seq = 2, data = m2>	→	A retransmite la expirarea intervalului de timp
10	←	<ack = 4, buf = 0>	←	Toate mesajele sunt confirmate, dar A este în continuare blocat
11	←	<ack = 4, buf = 1>	←	A poate să îl trimită acum pe 5
12	←	<ack = 4, buf = 2>	←	B a mai găsit un tampon
13	→	<seq = 5, data = m5>	→	A mai are un tampon liber
14	→	<seq = 6, data = m6>	→	A este blocat din nou
15	←	<ack = 6, buf = 0>	←	A este blocat în continuare
16	•••	<ack = 6, buf = 4>	←	Posibilă interblocare

Fig. 6-16. Alocarea dinamică a tampoanelor. Săgețile indică direcția transmisiei. Punctele de suspensie (...) indică pierderea unui TPDU.

Până acum am presupus tacit că singura limită impusă ratei de transfer a emițătorului este legată de dimensiunea spațiului alocat la receptor pentru tampoane. Deoarece prețul memoriei continuă să scadă vertiginos, ar putea deveni posibilă echiparea unei gazde cu suficient de multă memorie, astfel încât lipsa tampoanelor să pună rar probleme, dacă le va pune vreodată.

Atunci când spațiul de memorie alocat pentru tampoane nu limitează fluxul maxim, va apărea o altă limitare: capacitatea de transport a subrețelei. Dacă două rutere adiacente pot să schimbe cel mult x pachete pe secundă și există k căi distincte între două calculatoare gazdă, atunci este imposibil transferul la o rată mai mare de $k * x$ TPDU-uri pe secundă, oricât de multe tampoane ar fi alocate la cele două capete ale conexiunii. Dacă emițătorul se grăbește prea tare (adică trimite mai mult de $k * x$ TPDU-uri pe secundă), rețeaua se va congestiona, deoarece nu va putea să livreze datele le fel de repede cum le primește.

Este necesar un mecanism bazat pe capacitatea de transport a subrețelei și nu pe capacitatea de memorare în tampoane a receptorului. Evident, mecanismul de control al fluxului trebuie aplicat la emițător pentru a preveni existența prea multor TPDU-uri neconfirmate la acesta. Belsens (1975) a propus folosirea unei scheme cu fereastră glisantă în care emițătorul modifică dinamic dimensiunea ferestrei pentru a o potrivi la capacitatea de transport a rețelei. Dacă rețeaua poate să transporte c TPDU-uri pe secundă și durata unui ciclu (incluzând transmisia, propagarea, timpul petrecut în cozi, prelucrarea la destinație și revenirea confirmării) este r, atunci dimensiunea ferestrei la emițător trebuie să fie $c * r$. Folosind o fereastră cu această dimensiune, emițătorul va putea lucra la capacitate maximă. Orice mică scădere a performanțelor rețelei va genera blocări ale emițătorului.

Pentru a ajusta periodic dimensiunea ferestrei, emițătorul poate urmări cei doi parametri, după care poate calcula dimensiunea dorită a ferestrei. Capacitatea de transport a subrețelei poate fi de-

terminată pur și simplu numărând TPDU-urile confirmate într-o anumită perioadă de timp și raportând la acea perioadă. În timpul măsurătorii, emițătorul trebuie să transmită cât mai repede pentru a fi sigur că factorul care limitează numărul confirmărilor este capacitatea de transport a subrețelei și nu rata mică de emisie. Timpul necesar pentru ca un TPDU transmis să fie confirmat poate fi măsurat cu exactitate și media poate fi calculată continuu. Deoarece capacitatea de transport a rețelei disponibilă pentru orice flux dat variază în timp, dimensiunea ferestrei trebuie ajustată frecvent pentru a urmări schimbările în capacitatea de transport. Așa cum vom vedea mai departe, în Internet se folosește un mecanism similar.

6.2.5 Multiplexarea

Multiplexarea mai multor conversații pe conexiuni, circuite virtuale și legături fizice joacă un rol important în mai multe niveluri ale arhitecturii rețelei. În cazul nivelului transport, multiplexarea poate fi necesară din mai multe motive. De exemplu, dacă doar o singură adresă de rețea este disponibilă pe o gazdă, toate conexiunile transport de pe acea mașină trebuie să o folosească. Când un TDPU sosește este necesar un mod de a spune cărui proces trebuie dat. Această situație numită **multiplexare în sus**, este prezentată în fig. 6-17(a). În această figură, patru conexiuni transport diferite folosesc în comun aceeași conexiune rețea (de exemplu, adresa IP) către calculatorul gazdă de la distanță.

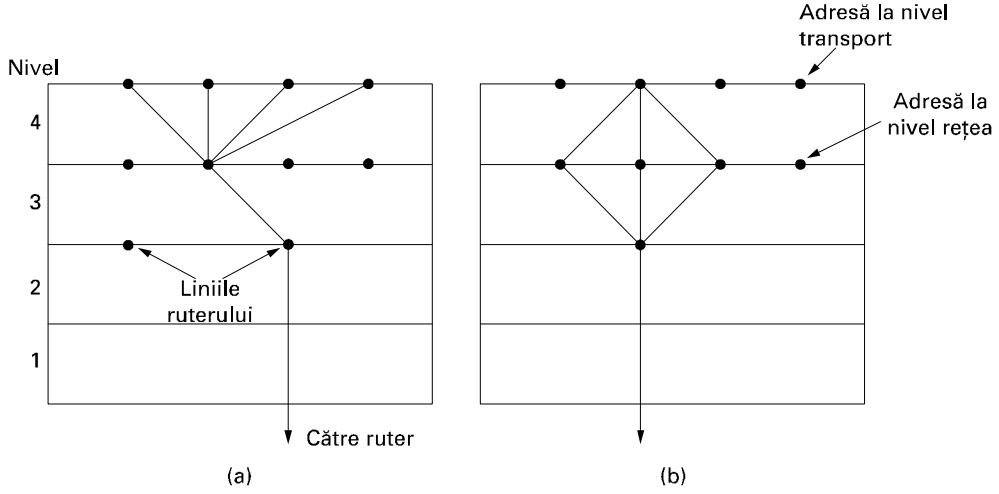


Fig. 6-17. (a) Multiplexare în sus. (b) Multiplexare în jos.

Multiplexarea poate să fie utilă nivelului transport și din alt motiv, legat de deciziile tehnice și nu de politica de prețuri ca până acum. Să presupunem, de exemplu, că o subrețea folosește intern circuite virtuale și impune rată de date maximă p fiecare dintre ele. Dacă un utilizator are nevoie de mai multă lățime de bandă decât poate oferi un circuit virtual, o soluție este ca nivelul transport să deschidă mai multe conexiuni rețea și să distribuie traficul prin acestea (într-un sistem round-robin), la fel ca în fig. 6-17(b). Acest mod de operare se numește **multiplexare în jos.** În cazul a *k* conexiuni rețea deschise lățimea de bandă reală este multiplicată cu un factor *k*. Un exemplu obișnuit de multiplexare în jos apare la utilizatorii care au acasă o linie ISDN. Această linie pune la dispoziție două conexiuni separate de 64 Kbps. Folosirea ambelor pentru apelul unui distribuitor de Internet și împărțirea traficului pe ambele linii, face posibilă atingerea unei lățimi de bandă efectivă de 128 Kbps.

6.2.6 Refacerea după cădere

În cazul în care calculatoarele gazdă sau ruterele se întâmplă să cadă, recuperarea după o astfel de cădere devine o problemă. Dacă entitatea de transport este în întregime conținută de calculatorul gazdă, atunci revenirea după o cădere a rețelei sau a unui ruter este simplă. Dacă nivelul rețea furnizează un serviciu datagramă, atunci entitatea de transport știe să rezolve problema TPDU-urilor pierdute. Dacă nivelul rețea furnizează un serviciu orientat pe conexiune, atunci pierderea unui circuit virtual este tratată stabilind un circuit virtual nou și apoi întrebând entitatea de transport aflată la distanță care TPDU-uri a primit deja și ce TPDU-uri nu. Acestea din urmă pot fi retransmise.

Căderea unui calculator gazdă pune o problemă mult mai supărătoare. În particular, clienții pot dori să continue să lucreze imediat după ce serverul cade și repornește. Pentru a ilustra această dificultate, să presupunem că o gazdă (clientul) trimite un fișier lung unei alte gazde (serverul) folosind un protocol simplu de tip pas-cu-pas (stop-and-wait). Nivelul transport de pe server nu face decât să paseze utilizatorului TPDU-urile primite, unul câte unul. Dacă în timpul transmisiei serverul cade, la revenirea acestuia tabelele sunt reinițializate și serverul nu va mai ști precis unde a rămas.

Într-o încercare de a reveni în starea sa inițială, serverul ar putea să trimită cereri tuturor celorlalte gazde anunțând că tocmai s-a refăcut după o cădere și cerând clienților să-l informeze despre situația tuturor conexiunilor deschise. Fiecare client poate fi în una din următoarele stări: un TPDU neconfirmat (starea S1) sau nici un TPDU neconfirmat (starea S0). Bazându-se numai pe această informație, clientul trebuie să decidă dacă să retransmită sau nu cel mai recent TPDU.

La prima vedere pare evident: atunci când află de căderea și revenirea serverului, clientul trebuie să retransmită doar dacă are TPDU-uri neconfirmate (adică este în starea S1). Totuși, o privire mai atentă descoperă dificultățile care apar în această abordare naivă. Să considerăm, ca exemplu, situația în care entitatea de transport de pe server trimite mai întâi confirmarea și apoi, după ce confirmarea a fost trimisă, pasează datele procesului aplicație. Trimiterea confirmării și transferul datelor procesului aplicație sunt două evenimente distincte care nu pot avea loc simultan. Dacă o cădere a serverului are loc după trimiterea confirmării, dar înainte de transferul datelor, clientul va primi confirmarea și se va afla în starea S0, atunci când anunțul despre cădere ajunge la el. În acest caz, clientul nu va mai retransmite (ceea ce este incorect), crezând că TPDU-ul respectiv a fost recepționat. Această decizie a clientului va conduce la lipsa unui TPDU.

În acest moment s-ar putea spune: „Nimic mai simplu! Tot ceea ce trebuie făcut este să reprogramăm entitatea de transport astfel, încât să transfere datele mai întâi și să trimită confirmarea după aceea!”. Să vedem: dacă transferul datelor a avut loc, dar serverul cade înainte să trimită confirmarea, clientul va fi în starea S1, va retransmite și astfel vom avea un TPDU duplicat în fluxul de date către procesul aplicație.

Oricum ar fi proiectați clientul și serverul, întotdeauna vor exista situații în care revenirea după o cădere nu se va face corect. Serverul poate fi programat în două feluri: să facă mai întâi confirmarea sau să transfere mai întâi datele. Clientul poate fi programat în patru feluri: să retransmită întotdeauna ultimul TPDU, să nu retransmită niciodată, să retransmită numai în starea S0, să retransmită numai în starea S1. Rezultă astfel opt combinații, dar, așa cum vom vedea, pentru fiecare combinație există o anumită succesiune de evenimente pentru care protocolul nu funcționează corect.

La server sunt posibile trei evenimente: trimiterea unei confirmări (A), transferul datelor la procesul aplicație (T) și căderea (C). Aceste trei evenimente pot să fie ordonate în 6 feluri: AC(T), ATC, C(AT), C(TA), TAC și TC(A). Parantezele sunt folosite pentru a indica faptul că nici A, nici

T nu pot urma după C (odată ce serverul a căzut, e bun căzut). Fig. 6-18 prezintă toate cele opt combinații ale strategiilor clientului și serverului și prezintă secvențele valide pentru fiecare din ele. Se observă că pentru orice strategie există o secvență de evenimente pentru care protocolul nu funcționează corect. De exemplu, dacă clientul retransmite întotdeauna, secvența ATC va genera un duplicat care nu poate fi detectat, în timp ce pentru celelalte două secvențe totul este în regulă.

	Strategia folosită de receptor					
	Mai întâi confirmă, apoi transferă			Mai întâi transferă, apoi confirmă		
Strategia folosită de emițător	AC(T)	ATC	C(AT)	C(TA)	T AC	TC(A)
Retransmite întotdeauna	OK	DUP	OK	OK	DUP	DUP
Nu retransmite niciodată	LOST	OK	LOST	LOST	OK	OK
Retransmite în S0	OK	DUP	LOST	LOST	DUP	OK
Retransmite în S1	LOST	OK	OK	OK	OK	DUP

OK = Protocolul funcționează corect
DUP = Protocolul generează un mesaj duplicat
LOST = Protocol pierde un mesaj

Fig. 6-18. Combinațiile diferitelor strategii posibile pentru server și client.

Încercarea de a reproiecta mai minuțios protocolul nu ajută. Chiar dacă clientul și serverul schimbă mai multe TPDU-uri înainte ca serverul să transfere datele, astfel încât clientul știe exact ceea ce se petrece pe server, nu poate afla dacă serverul a căzut imediat înainte sau imediat după transferul datelor. Concluzia se impune de la sine: dată fiind condiția de bază ca două evenimente să nu fie simultane, revenirea după o cădere nu poate fi făcută transparent pentru nivelurile superioare.

În termeni mai generali, acest rezultat poate fi reformulat astfel: restartarea după o cădere a nivelului N nu poate fi făcută decât de către nivelul N+1, și aceasta doar dacă nivelul superior reține suficientă informație de stare. Așa cum am arătat mai sus, nivelul transport poate să revină după erorile nivelului rețea numai dacă fiecare capăt al conexiunii ține minte unde a rămas.

Am ajuns astfel la problema definirii precise a ceea ce înseamnă o confirmare capăt-la-capăt. În principiu, protocolul de transport este unul capăt-la-capăt și nu înlănțuit ca la nivelurile de mai jos. Să considerăm cazul unui utilizator care generează cereri de tranzacții pentru o bază de date de la distanță. Să presupunem că entitatea de transport aflată la distanță este programată să trimită mai întâi TPDU-ul nivelului superior și apoi să confirme. Chiar și în acest caz, primirea unei confirmări de către mașina utilizator nu înseamnă neapărat că mașina gazdă de la distanță a avut timp să actualizeze baza de date. De fapt, o adevărată confirmare capăt-la-capăt, a cărei primire arată că s-au efectuat toate prelucrările de către mașina de la distanță, este probabil imposibil de obținut. Acest subiect este discutat în detaliu de Saltser ș.a. (1984).

6.3 UN PROTOCOL SIMPLU DE TRANSPORT

Pentru a concretiza ideile discutate, în această secțiune vom studia în detaliu un exemplu de nivel transport. Vom utiliza setul abstract de primitive orientate pe conexiune prezentat în fig. 6-2. Alegerea acestor primitive orientate pe conexiune face exemplul ales similar cu (dar mai simplu decât) popularul protocol TCP.

6.3.1 Primitivele serviciului ales ca exemplu

Prima problemă constă în definirea exactă a primitivelor de transport. Pentru CONNECT este ușor: vom avea o rutină de bibliotecă *connect* care poate fi apelată cu parametri potriviți pentru stabilirea unei conexiuni. Parametrii sunt TSAP-ul local și cel aflat la distanță. În timpul apelului, apelantul este blocat în timp ce entitatea de transport încearcă să stabilească conexiunea. Dacă conexiunea reușește, apelantul este deblocat și poate să înceapă să transmită date.

Atunci când un proces dorește să accepte conexiuni, el face un apel *listen* specificând un anumit TSAP pe care îl ascultă. După aceasta, procesul este blocat până când un proces aflat la distanță încearcă să stabilească o conexiune cu TSAP-ul la care așteaptă.

De observat că acest model este asimetric. O parte este pasivă, executând *listen* și așteptând ca ceva să se întâmple. Cealaltă parte este activă și inițiază conexiunea. O întrebare interesantă este: ce trebuie făcut dacă partea activă începe prima? O posibilitate este ca încercarea de conectare să nu reușească dacă nu există nici un proces care să asculte la TSAP-ul aflat la distanță. O altă posibilitate este ca inițiatorul să se blocheze (eventual pentru totdeauna) până când apare un proces care să asculte la TSAP-ul aflat la distanță.

Un compromis folosit în exemplul nostru este păstrarea cererii de conexiune la receptor pentru un anumit interval de timp. Dacă un proces de pe acest calculator gazdă apelează *listen* înainte ca timpul să expire, atunci conexiunea este stabilită, altfel conexiunea este refuzată și apelantul este deblocat întorcând un cod de eroare.

Pentru a elibera o conexiune vom folosi un apel *disconnect.* Atunci când ambele părți s-au deconectat, conexiunea este eliberată. Cu alte cuvinte, folosim un model de deconectare simetric.

Transmisia de date are exact aceeași problemă ca și stabilirea conexiunii: emițătorul este activ, dar receptorul este pasiv. Vom folosi aceeași soluție pentru transmisia de date ca și pentru stabilirea conexiunii, adică un apel activ *send* care trimite datele și un apel pasiv *receive* care blochează până când sosește un TPDU.

Definiția concretă a serviciului constă astfel din cinci primitive: CONNECT, LISTEN, DISCONNECT, SEND și RECEIVE. Fiecare primitivă corespunde unei funcții de bibliotecă care execută acea primitivă. Parametrii pentru primitive și funcțiile de bibliotecă sunt:

```
connum = LISTEN(local)
connum = CONNECT (local, remote)
status = SEND(connum, buffer, bytes)
status = RECEIVE(connum, buffer, bytes)
status = DISCONNECT(connum)
```

Primitiva LISTEN anunță disponibilitatea serverului de a accepta cereri de conexiune la TSAP-ul indicat. Utilizatorul primitivei este blocat până când se face o încercare de conectare la TSAP-ul specificat. Blocarea poate să fie definitivă.

Primitiva CONNECT are doi parametri, un TSAP local (adică o adresă la nivel transport) și un TSAP aflat la distanță și încearcă să stabilească o conexiune între acestea două. Dacă reușește, ea întoarce *connum,* un număr nenegativ utilizat pentru a identifica conexiunea. Dacă nu reușește, motivul este pus în *connum* ca un număr negativ. În modelul nostru (destul de simplu), fiecare TSAP poate să participe la cel mult o singură conexiune de transport, deci un motiv pentru refuzul unei cereri de conectare poate fi că adresa la nivel transport este deja folosită. Alte câteva motive pot fi: gazda de la distanță căzută, adresă locală incorectă sau adresă de la distanță incorectă.

Primitiva SEND trimite conținutul unui tampon ca un mesaj pe conexiunea indicată, eventual divizându-l în mai multe unități, dacă este nevoie. Erorile posibile, întoarse în *status*, pot fi: nu există conexiune, adresă de tampon invalidă sau număr de biți negativ.

Primitiva RECEIVE indică faptul că apelantul așteaptă să recepționeze date. Dimensiunea mesajului este plasată în câmpul *bytes.* Dacă procesul aflat la distanță a eliberat conexiunea sau dacă adresa pentru tampon este incorectă (de exemplu, în afara spațiului de adrese al programului utilizator), funcția va întoarce un cod de eroare indicând natura problemei.

Primitiva DISCONNECT pune capăt unei conexiunii transport indicate prin parametrul *connum*. Erori posibile sunt: *connum* aparține de fapt altui proces sau *connum* nu este un identificator valid de conexiune. Codul de eroare sau 0 pentru succes sunt returnate în *status.*

6.3.2 Entitatea de transport aleasă ca exemplu

Înainte de a studia programul aferent entității de transport, trebuie spus că acesta este un exemplu similar celor din cap. 3: este prezentat mai mult în scop pedagogic decât pentru a fi utilizat. Mai multe detalii tehnice (precum detectarea extensivă a erorilor) care ar fi necesare unui produs real au fost lăsate deoparte pentru simplitate.

Nivelul transport utilizează primitivele serviciului rețea pentru a trimite și recepționa TPDU-uri. Trebuie să alegem primitivele serviciului rețea pe care le vom utiliza pentru acest exemplu. O posibilitate ar fi fost: un serviciu datagramă nesigur. Pentru a păstra simplitatea exemplului, nu am făcut această alegere. Folosind un serviciu datagramă nesigur, codul pentru entitatea de transport ar fi devenit foarte mare și complicat, în cea mai mare parte legat de pachete pierdute sau întârziate. Și în plus, cea mai mare parte a acestor idei au fost deja discutate în cap. 3.

Am ales în schimb un serviciu rețea sigur, orientat pe conexiune. În acest fel ne putem concentra asupra problemelor puse de nivelul transport care nu apar în nivelurile inferioare. Acestea includ, între altele, stabilirea conexiunii, eliberarea conexiunii și gestiunea tampoanelor. Un serviciu de transport simplu, construit peste un nivel rețea ATM, ar putea să semene cu acesta.

În general, entitatea de transport poate să fie parte a sistemului de operare al calculatorului gazdă sau poate să fie un pachet de funcții de bibliotecă în spațiul de adrese utilizator. Pentru simplitate, exemplul nostru a fost programat ca și cum entitatea de transport ar fi un pachet de funcții de bibliotecă, dar schimbările necesare pentru a o face parte a sistemului de operare sunt minime (fiind în principal legate de modul cum sunt adresate tampoanele).

Totuși merită remarcat faptul că, în acest exemplu, „entitatea de transport” nu este o entitate separată, ci este o parte a procesului utilizator. Mai precis, atunci când utilizatorul folosește o primitivă blocantă, de exemplu LISTEN, se blochează toată entitatea de transport. În timp ce această arhitectură este potrivită pentru un calculator gazdă cu un singur proces utilizator, pentru un calculator gazdă cu mai mulți utilizatori este normal ca entitatea de transport să fie un proces separat, distinct de toate celelalte procese.

Interfața cu nivelul rețea se face prin intermediul procedurilor *to_net* și *from_net.* Fiecare are șase parametri. Primul este identificatorul conexiunii, care este în corespondență bijectivă cu circuitul virtual la nivel rețea. Apoi urmează biții *Q* și *M* care, atunci când au valoarea 1, indică mesaje de control și, respectiv, faptul că mesajul continuă și în următorul pachet. După acestea urmează tipul pachetului, ales dintr-un set de șase tipuri de pachete prezentate în fig. 6-19. La sfârșit se găsește un indicator către zona de date și un întreg care indică numărul de octeți de date.

Tip pachet	Explicații
CALL_ REQUEST	Trimis pentru a stabili conexiunea
CALL_ACCEPTED	Răspuns la CALL_ REQUEST
CLEAR_REQUEST	Trimis pentru a elibera conexiunea
CLEAR_CONFIRMATION	Răspuns la CLEAR_REQUEST
DATA	Pentru transport de date
CREDIT	Pachet de control pentru gestionarea ferestrei

Fig. 6-19. Tipurile de pachete folosite la nivel rețea.

La apelurile *to_net,* entitatea de transport completează toți parametrii pentru ca nivelul rețea să-i poată citi; la apelurile *from_net* nivelul rețea dezasamblează un pachet sosit pentru entitatea de transport. Transferul informației sub forma unor parametri ai unei proceduri și nu a pachetului de trimis sau de recepționat protejează nivelul transport de toate detaliile protocolului nivelului rețea. Dacă entitatea de transport încearcă să trimită un pachet atunci când fereastra glisantă corespunzătoare a circuitului virtual este plină, ea va fi blocată într-un apel *to_net* până când va fi spațiu în fereastră. Mecanismul este transparent pentru entitatea de transport și este controlat de nivelul rețea prin comenzi ca *enable_transport_layer* și *disable_transport_layer,* analoage celor descrise în protocoalele din cap. 3. Gestionarea ferestrei de pachete este de asemenea făcută de către nivelul rețea.

Pe lângă acest mecanism transparent de suspendare mai există două proceduri (care nu sunt prezentate aici), *sleep* și *wakeup,* apelate de entitatea de transport. Procedura *sleep* este apelată atunci când entitatea de transport este blocată logic în așteptarea unui eveniment extern, în general sosirea unui pachet. După ce a fost apelat *sleep*, entitatea de transport (și procesul utilizator, bineînțeles) sunt blocate.

Codul entității de transport este prezentat în fig. 6-20. Orice conexiune este într-una din următoarele șapte stări:

1. *IDLE* - conexiunea nu a fost încă stabilită
2. *WAITING* - a fost executat CONNECT și trimis: CALL_REQUEST
3. *QUEUED* - a sosit un CALL_REQUEST, nu a fost executat încă nici un apel LISTEN
4. *ESTABLISHED* - conexiunea a fost stabilită
5. *SENDING* - utilizatorul așteaptă permisiunea de a trimite un pachet
6. *RECEIVING* - a fost apelat RECEIVE
7. *DISCONNECTING* - un apel DISCONNECT a fost făcut local

Pot să apară tranziții între stări ori de câte ori are loc unul din următoarele evenimente: este executată o primitivă, sosește un pachet sau expiră un interval de timp stabilit.

Procedurile prezentate în fig. 6-20 sunt de două tipuri. Majoritatea sunt apelate direct de către programele utilizator, totuși *packet_arrival* și *clock* sunt diferite. Execuția lor este declanșată de evenimente externe: sosirea unui pachet și, respectiv, avansul ceasului. Ele sunt de fapt rutine de întrerupere. Vom presupune că acestea nu sunt niciodată apelate atât timp cât rulează o altă procedură a entității de transport. Acestea pot fi executate numai atunci când procesul utilizator este în așteptare sau atât timp cât el rulează în afara entității de transport. Această proprietate este crucială pentru funcționarea corectă a entității de transport.

```
#define MAX_CONN 32                    /* numărul maxim de conexiuni deschise simultan */
#define MAX_MSG_SIZE 8192              /* dimensiunea maximă a unui mesaj în octeți */
#define MAX_PKT_SIZE 512               /* dimensiunea maximă a unui pachet în octeți */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT,CREDIT}
pkt_type;
typedef enum {IDLE, WAITING, QUEUED, ESTABLISHED, SENDING, RECEIVING, DISCONN}
cstate;

/* Variabile globale */
transport_address listen_address;              /* adresa locală care este ascultată */
int listen_conn;                               /* identificatorul de conexiune pentru listen */
unsigned char data[MAX_PKT_SIZE]               /* zona pentru pachetele de date */

struct conn {
    transport_address local_address, remote_address;
    cstate state;                                          /* starea conexiunii */
    unsigned char *user_buf_addr;                          /* pointer la tamponul de recepție */
    int byte_count;                                        /* contor de emisie/recepție*/
    int clr_req_received;                  /* setat atunci când este primit un pachet CLEAR_REQ*/
    int timer;                             /* folosit pentru a evita așteptările infinite */
    int credits;                           /* numărul de mesaje care poate fi trimis */
}; conn[MAX_CONN + 1]                                      /* poziția 0 nu e folosită */

/* prototipuri */
void sleep(void);
void wakeup(void);
void to_net(int cid, int q, int *m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt,unsigned char *p, int *bytes);

int listen (transport_address t)
{
/* Utilizatorul vrea stabilească o conexiune. Trebuie văzut daca CALL_REQ a sosit deja */
    int i, found=0;

    for (i=1; i<= MAX_CONN; i++)               /* caută în tabela de conexiuni CALL_REQ*/
        if (conn[i].state == QUEUED && conn[i].locall_address == t) {
            found = i;
            break;
        }
        if (found == 0) {          /* nu a găsit nici un CALL_REQ. Așteaptă până când sosește
                                      unul sau până când expira intervalul de timp de așteptare */
            listen_address = t; sleep(); i = listen_conn;
        }
    conn[i].state = ESTABLISHED;                           /* conexiunea este stabilită */
```

```
    conn[i].timer = 0                                   /* ceasul nu este folosit */
    listen_conn = 0;                                    /* 0 este presupus a fi o adresă invalidă */
    to_net(i, 0, 0, CALL_ACC, 0);          /* spune niv. rețea să accepte cererea de conexiune */
    return(i);                                          /* întoarce identificatorul conexiunii */
}

int connect (transport_address l, transport_address r)
{                              /* Utilizatorul dorește să stabilească o conexiune cu un proces aflat
                                                                  la distanță; se trimite un pachet CALL_REQ */
    int i;
    struct conn *cptr;

    data[0] = r;                                /* pachetul CALL_REQ are nevoie de acestea */
    data[1] = l;
    i = MAX_CONN;                                                 /* caută în tabelă înapoi */
    while (conn[i].state != IDLE && i>1) i = i-1;
    if (conn[i].state == IDLE) {                                  /* Face o intrare în tabelă */
        cptr = &conn[i];
        cptr->local_address = l;;
        cptr->remote_address = r;
        cptr->state = WAITING;
        cptr->clr_req_received = 0;
        cptr->credits = 0;
        cptr->timer = 0;
        to_net(i, 0, 0, CALL_REQ, data, 2);
        sleep();                                    /* așteaptă CALL_ACC sau CLEAR_REQ */
        if (cptr->state == ESTABLISHED) return (i);
        if (cptr->clr_req_received ) {              /* cererea de conexiune este refuzată */
            cptr->state = IDLE                                    /* înapoi în starea IDLE */
            to_net(i, 0, 0, CLEAR_CONF, data, 0);
            return(ERR_REJECT);
        }
    }
    else return (ERR_FULL)          /* conexiunea este refuzată: insuficient spațiu în tabele */
}

int send (int cid, unsigned char bufptr[], int bytes)
{                                                       /* Utilizatorul dorește să trimită un mesaj */
    int i, count, m;
    struct conn *cptr = &conn[cid];

    /* Intră în starea SENDING */
    cptr->state = SENDING;
    cptr->byte_count = 0;
    if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
    if (cptr->clr_req_received == 0) {
                                  /* Există credite; împarte mesajul în pachete dacă este cazul */
        do {
            if (bytes - cptr->byte_count > MAX_PKT_SIZE) {
                /* mesaj format din mai multe pachete */
                count = MAX_PKT_SIZE;
                m = 1;
                /* mai urmează pachete */
```

```
            } else { /* un mesaj format dintr-un pachet */
                count = bytes — cptr->byte->count;
                m=0;                                    /* ultimul pachet al acestui mesaj */
            }
            for (i=0; i<count; i++) data[i]= bufptr[cptr->byte_count+1];
            to_net(cid, 0, m,DATA_PKT, data, count);          /* trimite un pachet */
            cptr->byte_count=cptr->byte_count + count;   /* incrementează nr. octeților trimiși */
        } while (cptr->byte_count < bytes);  /* ciclează până când întregul mesaj este trimis */
        cptr->credits--;                                /* fiecare mesaj folosește un credit */
        cptr->state = ESTABLISHED;
        return(OK);
    } else { cptr->state = ESTABLISHED;
        return(ERR_CLOSED);     /* întoarce insucces: celălalt capăt vrea să se deconecteze */
    }
}

int receive (int cid, unsigned char bufptr[], int *bytes)
{
    /* Utilizatorul este gata să primească un mesaj */
    struct conn *cptr = &conn[cid];

    if (cptr->clr_req_received == 0) {   /* Conexiunea încă stabilită; încearcă să recepționeze */
        cptr->state = RECEIVING;
        cptr->user_buf_addr = bufptr;
        cptr->byte_count = 0;
        data[0] = CRED;
        data[1] = 1;
        to_net(cid, 1, 0, CREDIT, data, 2);                    /* trimite CREDIT */
        sleep();                                  /* se blochează în așteptarea datelor */
        *bytes = cptr->byte_count;
    }
    cptr->state = ESTABLISHED;
    return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect (int cid)
{   /* Utilizatorul vrea să se deconecteze */
    struct conn *cptr = &conn[cid];

    if (cptr -> clr_req_received) {                /* cealaltă parte a inițiat deconectarea */
        cptr->state = IDLE;                                  /* conexiunea este eliberată */
        to_net(cid, 0, 0, CLEAR_CONF, data, 0);
    } else {                                                  /* se inițiază terminarea */
        cptr->state = DISCONNECT;                /* conexiunea nu este eliberată până când
                                                              cealaltă parte nu-și dă acordul */
        to_net(cid, 0, 0, CLEAR_REQ, data, 0);
    }
    return (OK);
}

void packet_arrival (void)
{   /* A sosit un pachet; urmează prelucrarea lui */
    int cid;                                    /* conexiunea pe care a sosit pachetul */
```

```
    int count, i q, m;
    pkt_type ptype;                                   /* CALL_REQ, CALL_ACC, CLEAR_REQ,
                                                        CLEAR_CONF, DATA_PKT, CREDIT */
    unsigned char data [MAX_MKT_SIZE];                /* date din pachetul care sosește */
     struct conn *cptr;

    from_net(&cid, &q, &ptype, data, &count);                          /* preia pachetul */
    cptr = &conn[cid];
    switch (ptype) {
        case CALL_REQ:        /* utilizatorul de la distanță vrea să stabilească o conexiune */
            cptr->local_address = data[0];
            cptr->remote_address = data[1];
            if (cptr->local_address == listen_address) {
                listen_conn = cid;
                cptr->state = ESTABLISHED;
                wakeup();
            } else {
                cptr->state = QUEUED;
                cptr->timer = TIMEOUT;
            }
            cptr->clr_req_received = 0;
            cptr->credits = 0;
            break;

        case CALL_ACC:            /* utilizatorul de la distanță a acceptat CALL_REQ trimis */
            cptr->state = ESTABLISHED;
            wakeup();
            break;

        case CLEAR_REQ:
                  /* utilizatorul de la distanță vrea să se deconecteze sau să refuze un apel */
            cptr->clr_req_received = 1;
            if (cptr->state == DISCONN) cptr->state = IDLE;
            if (cptr->state == WAITING || cptr->state == RECEIVING
                        || cptr->state == SENDING) wakeup();
            break;

        case CLEAR_CONF:                /* utilizatorul de la distanță acceptă deconectarea */
            cptr->state = IDLE;
            break;

        case CREDIT:                                  /* utilizatorul de la distanță așteaptă date */
            cptr->credits +=data[1];
             if (cptr->state == SENDING) wakeup();
             break;

        case DATA_PKT:                                                    /* au fost trimise date */
            for (i=0; i<count; i++)
                cptr->user_buff_addr[cptr->byte_count + i] = data[i];
            cptr->byte_count += count;
            if (m == 0) wakeup();
    }
}
```

```
void clock( void)
{                   /* la fiecare interval de timp al ceasului se verifică eventualele
                       depășiri ale timpilor de așteptare pentru cererile aflate în starea QUEUED */
     int i;
     struct conn *cptr;

     for (i=1; i<=MAX_CONN, i++) {
          cptr = &conn[i];
          if (cptr->timer > 0) {                                    /* ceasul funcționa */
               cptr->timer --;
               if (cptr->timer == 0) {                              /* timpul a expirat */
                    cptr->state = IDLE;
                    to_net(i, 0, 0, CLEAR_REQ, data, 0);
               }
          }
     }
}
```

Fig. 6-20. Entitatea de transport aleasă ca exemplu.

Existența bitului *Q* în antetul pachetului ne permite să evităm timpul suplimentar introdus de antetul protocolului de transport. Mesajele de date obișnuite sunt trimise ca pachete de date cu $Q=0$. Mesajele legate de protocolul de transport, dintre care există numai unul (CREDIT) în exemplul nostru, sunt trimise ca pachete de date cu $Q=1$. Aceste mesaje de control sunt detectate și prelucrate de entitatea de transport receptoare.

Structura de date de bază folosită de entitatea de transport este vectorul *conn,* care are câte o înregistrare pentru fiecare conexiune posibilă. În înregistrare sunt menținute starea conexiunii, incluzând adresa de nivel transport la fiecare capăt, numărul de mesaje trimise și recepționate pe conexiune, starea curentă, un indicator (pointer) la tamponul utilizator, numărul de octeți ai mesajului trimis sau recepționat, un bit indicând dacă utilizatorul aflat la distanță a apelat DISCONNECT, un ceas, și un contor folosit pentru a permite transmiterea mesajelor. Nu toate aceste câmpuri sunt folosite în exemplul nostru simplu, dar o entitate de transport completă ar avea nevoie de toate, poate chiar de mai multe. Fiecare element din *conn* este inițializat cu starea *IDLE* .

Atunci când un utilizator apelează CONNECT, nivelului rețea i se va cere să trimită un pachet CALL_REQUEST către mașina de la distanță și utilizatorul este blocat. Atunci când pachetul CALL_REQUEST ajunge de partea cealaltă, entitatea de transport este întreruptă pentru a executa *packet_arrival,* care verifică dacă utilizatorul local ascultă la adresa specificată. Dacă da, atunci este trimis înapoi un pachet CALL_ACCEPTED și utilizatorul de la distanță este trezit; dacă nu, atunci cererea CALL_REQUEST este pusă într-o coadă pentru un interval de timp TIMEOUT. Dacă în acest interval este făcut un apel LISTEN, atunci conexiunea este stabilită; dacă nu, conexiunea este refuzată la sfârșitul intervalului de timp cu un pachet CLEAR_REQUEST ca să nu fie blocată pe timp nedefinit.

Deși am eliminat antetul pentru protocolul de transport, tot mai trebuie găsită o modalitate pentru a determina cărei conexiuni transport îi aparține un anumit pachet, deoarece pot exista simultan mai multe conexiuni. Cea mai simplă soluție este folosirea numărului de circuit virtual de la nivel rețea și ca număr de conexiune transport. În plus, numărul de circuit virtual poate fi folosit și ca index în vectorul *conn.* Atunci când un pachet sosește pe circuitul virtual *k* la nivel rețea, el îi va aparține conexiunii *k* a cărei stare este păstrată în *conn[k].* La inițierea unei conexiuni, numărul de cone-

xiune este ales de entitatea de transport care a inițiat-o. Pentru cererile de conexiune, nivelul rețea alege ca număr de conexiune orice număr de circuit virtual care nu a fost încă folosit.

Pentru a evita gestionarea tampoanelor în interiorul entității de transport, este folosit un mecanism de gestiune a fluxului diferit de fereastra glisantă tradițională. Când un utilizator apelează RECEIVE, este trimis un **mesaj de credit** entității de transport de pe mașina care va transmite și este înregistrat în vectorul *conn.* Atunci când este apelat SEND, entitatea de transport verifică dacă a sosit vreun credit pe conexiunea respectivă. Dacă da, mesajul este trimis (chiar și în mai multe pachete, dacă este cazul) și credit este decrementat; dacă nu, atunci entitatea de transport se blochează până la sosirea unui credit. Mecanismul garantează că nici un mesaj nu este trimis decât dacă cealaltă parte a apelat deja RECEIVE. Ca rezultat, ori de câte ori sosește un mesaj, există cu siguranță un tampon disponibil pentru el. Această schemă poate fi ușor generalizată pentru a permite receptorilor să ofere tampoane multiple și să ceară mai multe mesaje.

Trebuie reținută simplitatea codului din fig. 6-20. O entitate de transport reală ar verifica în mod normal validitatea tuturor parametrilor furnizați de utilizator, ar putea să revină după căderea rețelei, ar putea rezolva coliziunile la apel și ar susține un serviciu transport mult mai general, care ar include facilități cum sunt întreruperile, datagramele și versiunile nonblocante ale primitivelor SEND și RECEIVE.

6.3.3 Exemplul văzut ca un automat finit

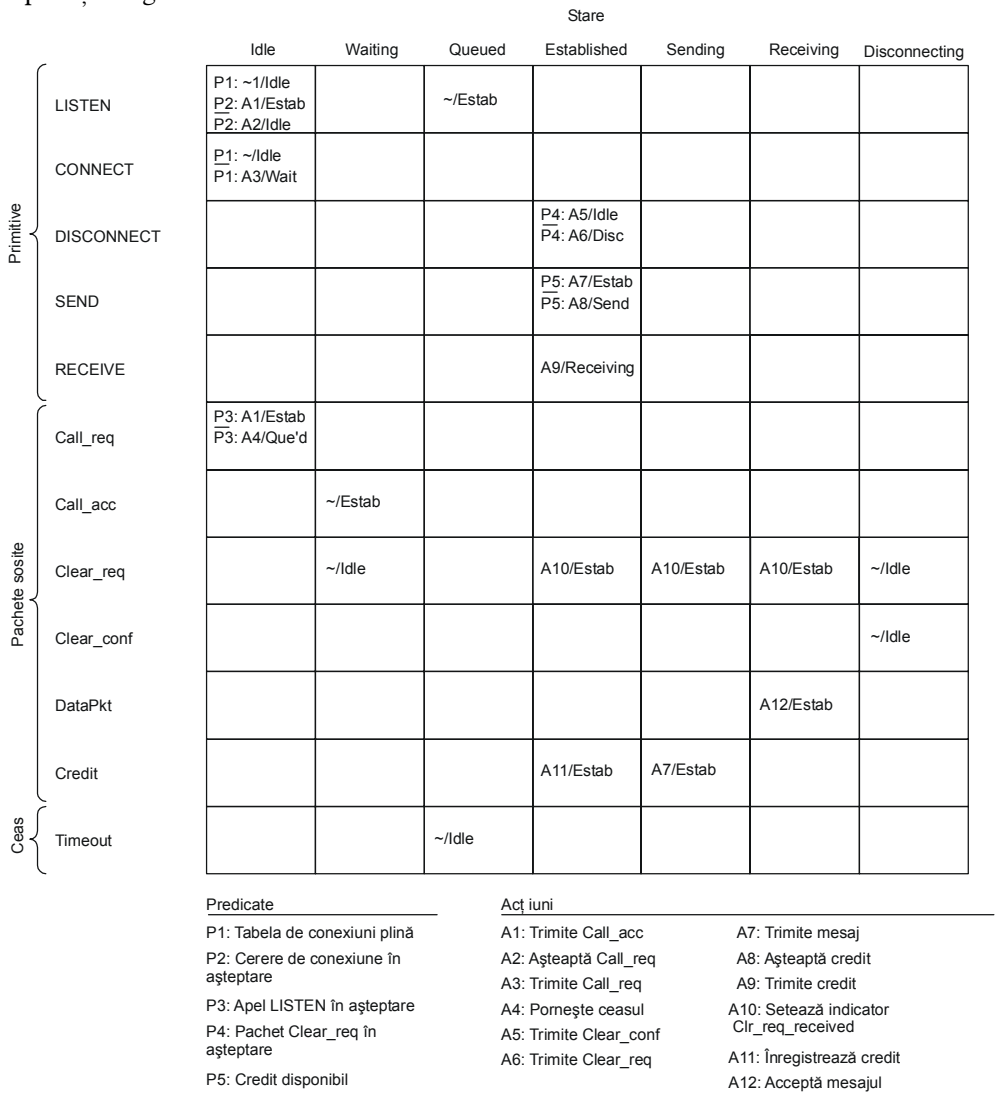
Scrierea unei entități de transport este o muncă dificilă și riguroasă, în special pentru protocoalele reale. Pentru a reduce probabilitatea de apariție a unei erori, adeseori este util să reprezentăm protocolul ca un automat finit.

Am văzut deja că protocolul nostru folosit ca exemplu are șapte stări pentru fiecare conexiune. Este de asemenea posibil să izolăm cele douăsprezece evenimente care pot să apară pentru a schimba starea unei conexiuni. Cinci dintre aceste evenimente sunt cele cinci primitive ale serviciului de transport. Alte șase sunt reprezentate de sosirile celor șase tipuri distincte de pachete. Ultimul este expirarea intervalului de timp stabilit. Fig. 6-21 arată acțiunile principale ale protocolului sub forma unei matrice. Pe coloane sunt prezentate stările, iar pe linii cele 12 evenimente.

Fiecare intrare în matrice (adică în automatul finit) din fig. 6-21 are până la trei câmpuri: un predicat, o acțiune și o nouă stare. Predicatul indică condițiile în care acțiunea este executată. De exemplu, pentru intrarea din colțul stânga-sus, dacă este executat un LISTEN și nu mai există spațiu în tabele (predicatul P1), atunci LISTEN eșuează și starea rămâne aceeași. Pe de altă parte, dacă un pachet CALL_REQUEST a sosit deja la adresa de nivel transport la care se face LISTEN (predicatul P2), atunci conexiunea este stabilită imediat. O altă posibilitate este ca P2 să fie fals, adică nici un CALL_REQUEST nu a fost primit, caz în care conexiunea rămâne în starea *IDLE* în așteptarea unui pachet CALL_REQUEST.

Merită să subliniem că alegerea stărilor folosite în matrice nu este în întregime determinată de protocolul însuși. În acest exemplu, nu există nici o stare *LISTENING*, care ar fi putut urma în mod normal apelului LISTEN. Nu a fost introdusă o stare *LISTENING* deoarece o stare este asociată cu o intrare în tabela de conexiuni și la un apel LISTEN nu se creează nici o intrare în tabelă. De ce? Pentru că am decis să folosim identificatorii circuitelor virtuale de la nivel rețea ca identificatori pentru conexiune și, pentru un apel LISTEN, numărul circuitului virtual este în cele din urmă ales de nivelul rețea atunci când sosește un CALL_REQUEST.

Acțiunile de la A1 la A12 sunt acțiuni importante, precum trimiterea pachetelor sau rearmarea ceasurilor. Nu sunt menționate toate acțiunile minore, cum ar fi inițializarea unor câmpuri ale înregistrării atașate conexiunii. Dacă o acțiune implică trezirea unui proces în așteptare, atunci acțiunile care urmează trezirii procesului sunt considerate și ele. De exemplu, dacă un pachet CALL_REQUEST sosește și există un proces adormit care îl așteaptă, atunci transmiterea pachetului CALL_ACCEPT este considerată ca făcând parte din acțiunea care urmează recepției lui CALL_REQUEST. După ce este efectuată fiecare acțiune, conexiunea ajunge într-o nouă stare, așa cum apare și în fig. 6-21.



		Stare: Idle	Waiting	Queued	Established	Sending	Receiving	Disconnecting
Primitive	LISTEN	P1: ~1/Idle P2: A1/Estab P2: A2/Idle		~/Estab				
	CONNECT	P1: ~/Idle P1: A3/Wait						
	DISCONNECT				P4: A5/Idle P4: A6/Disc			
	SEND				P5: A7/Estab P5: A8/Send			
	RECEIVE				A9/Receiving			
Pachete sosite	Call_req	P3: A1/Estab P3: A4/Que'd						
	Call_acc		~/Estab					
	Clear_req		~/Idle		A10/Estab	A10/Estab	A10/Estab	~/Idle
	Clear_conf							~/Idle
	DataPkt						A12/Estab	
	Credit				A11/Estab	A7/Estab		
Ceas	Timeout			~/Idle				

Predicate

P1: Tabela de conexiuni plină

P2: Cerere de conexiune în așteptare

P3: Apel LISTEN în așteptare

P4: Pachet Clear_req în așteptare

P5: Credit disponibil

Acțiuni

A1: Trimite Call_acc

A2: Așteaptă Call_req

A3: Trimite Call_req

A4: Pornește ceasul

A5: Trimite Clear_conf

A6: Trimite Clear_req

A7: Trimite mesaj

A8: Așteaptă credit

A9: Trimite credit

A10: Setează indicator Clr_req_received

A11: Înregistrează credit

A12: Acceptă mesajul

Fig. 6-21. Protocolul ales ca exemplu, reprezentat ca un automat finit. Fiecare intrare are un predicat opțional, o acțiune opțională și o nouă stare. Caracterul tilda indică faptul că nici o acțiune importantă nu este efectuată. Bara deasupra predicatului este reprezentarea pentru predicatul negat. Intrările vide corespund unor secvențe de evenimente imposibile sau eronate.

Avantajul reprezentării protocolului ca o matrice este întreit. În primul rând, în această formă este mult mai simplu pentru programator să verifice sistematic fiecare combinație stare/ eveniment/ acțiune. În implementările reale, unele combinații vor fi folosite pentru tratarea erorilor. În fig. 6-21 nu s-a făcut nici o distincție între situațiile imposibile și cele care sunt numai ilegale. De exemplu, dacă o conexiune este în starea *WAITING*, evenimentul DISCONNECT este imposibil deoarece utilizatorul este blocat și nu poate să facă nici un apel. Pe de altă parte, în starea *SENDING* nu pot sosi date deoarece nu a fost generat nici un credit. Sosirea unui pachet de date va fi o eroare a protocolului.

Al doilea avantaj al reprezentării protocolului ca o matrice este legat de implementarea acestuia. Într-un vector bidimensional elementul *act[i][j]* ar putea fi văzut ca un indicator la procedura care tratează evenimentul i atunci când starea este j. O implementare posibilă este codificarea entității de transport ca o buclă în care se așteaptă la început producerea unui eveniment. După producerea evenimentului, este identificată conexiunea implicată și este extrasă starea ei. Cunoscând acum starea și evenimentul, entitatea de transport nu face decât să acceseze vectorul *act* și să apeleze procedura adecvată. Această abordare ar conduce la o arhitectură mult mai regulată și mai simetrică decât cea a entității de transport implementate de noi.

Al treilea avantaj al abordării folosind un automat finit este legat de descrierea protocolului. În unele standarde, protocoalele sunt specificate ca un automat finit de tipul celui din fig. 6-21. Trecerea de la acest tip de descriere la o entitate de transport funcțională este mult mai ușoară dacă entitatea de transport este și ea modelată cu ajutorul unui automat finit.

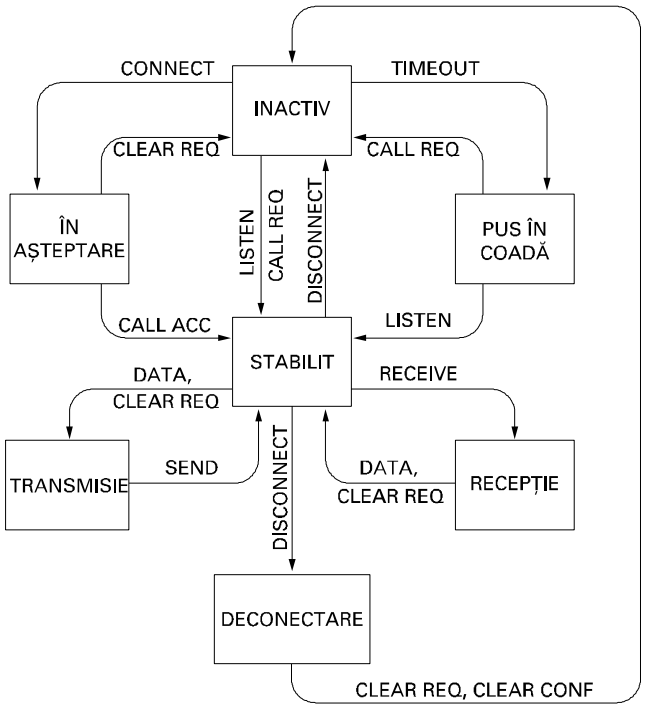


Fig. 6-22. Protocolul dat ca exemplu, în reprezentare grafică. Pentru simplificare, tranzițiile care lasă starea conexiunii nemodificată au fost omise.

Cel mai important dezavantaj este că această abordare poate să fie mai dificil de înțeles decât metoda directă pe care am utilizat-o la început. Totuși, această problemă poate fi rezolvată, fie și parțial, desenând automatul finit ca un graf, așa cum am făcut-o în fig. 6-22.

6.4 PROTOCOALE DE TRANSPORT PRIN INTERNET: UDP

Internet-ul are două protocoale principale în nivelul de transport: unul neorientat pe conexiune și unul orientat pe conexiune. În următoarele secțiuni o să le studiem pe ambele. Protocolul neorientat pe conexiune se numește UDP. Protocolul orientat pe conexiune se numește TCP. O să începem cu UDP-ul deoarece în esență este la fel ca IP-ul cu un mic antet adăugat. De asemeni, o să studiem și două aplicații ale UDP-ului.

6.4.1 Introducere în UDP

Setul de protocoale Internet suportă un protocol de transport fără conexiune, **UDP** (**User Protocol** – Protocol cu Datagrame Utilizator). UDP oferă aplicațiilor o modalitate de a trimite datagrame IP încapsulate și de a le transmite fără a fi nevoie să stabilească o conexiune. UDP este descris în RFC 768.

UDP transmite **segmente** constând într-un antet de 8 octeți urmat de informația utilă . Antetul este prezentat în fig. 6-23. Cele două porturi servesc la identificarea punctelor terminale ale mașinilor sursă și destinație. Când ajunge un pachet UDP, conținutul său este predat procesului atașat portului destinație. Această atașare are loc atunci când se folosește o simplă procedură de nume sau ceva asemănător, așa cum am văzut în fig. 6-6 pentru TCP (procesul de legătură este același pentru UDP). De fapt, valoarea cea mai importantă dată de existența UDP-ului față de folosirea doar a IP-ului simplu, este aceea a adăugării porturilor sursă și destinație. Fără câmpurile portului, nivelul de transport nu ar ști ce să facă cu pachetul. Cu ajutorul lor, segmentele se livrează corect.

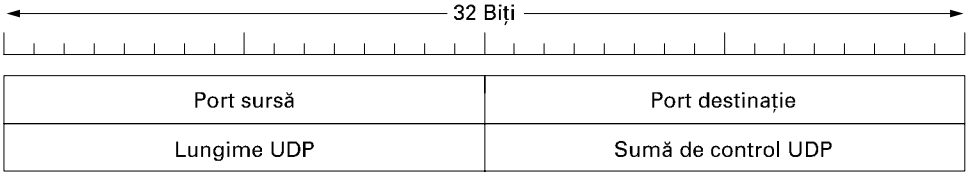


Fig. 6-23. Antetul UDP.

Portul sursă este în primul rând necesar atunci când un răspuns trebuie transmis înapoi la sursă. Prin copierea câmpului *port sursă* din segmentul care sosește în câmpul *port destinație* al segmentului care pleacă, procesul ce trimite răspunsul specifică ce proces de pe mașina de trimitere urmează să-l primească.

Câmpul l*ungime UDP* include antetul de 8 octeți și datele. Câmpul *sumă de control UDP* este opțional și stocat ca 0 (zero) dacă nu este calculat (o valoare de adevăr 0 rezultată în urma calculelor

este memorată ca un șir de biți 1). Dezactivarea acestuia este o prostie, excepție făcând cazul în care calitatea informației chiar nu contează (de exemplu, transmisia vocală digitalizată).

Merită probabil menționate, în mod explicit, unele dintre lucrurile pe care UDP-ul *nu le face*. Nu realizează controlul fluxului, controlul erorii, sau retransmiterea unui segment incorect primit. Toate acestea depind de procesele utilizatorului. Ceea ce face este să ofere protocolului IP o interfață cu facilități adăugate de demultiplexare a mai multor procese, folosind porturi. Aceasta este tot ceea ce face UDP-ul. Pentru aplicațiile care trebuie să aibă un control precis asupra fluxului de pachete, controlului erorii sau cronometrarea, UDP-ul furnizează doar ceea ce "a ordonat doctorul".

Un domeniu unde UDP-ul este în mod special util este acela al situațiilor client-server. Deseori, clientul trimite o cerință scurtă server-ului și așteaptă înapoi un răspuns scurt. Dacă se pierde ori cererea ori răspunsul, clientul poate pur și simplu să încerce din nou după ce a expirat timpul. Nu numai că va fi mai simplu codul, dar sunt necesare și mai puține mesaje (câte unul în fiecare direcție) decât la un protocol care solicită o inițializare inițială.

O aplicație care folosește UDP-ul în acest fel este DNS (Domain Name System, rom: Sistem de rezolvare de nume), pe care îl vom studia în cap. 7. Pe scurt, un program care trebuie să caute adresele de IP ale unor nume gazdă, de exemplu *www.cs.berkley.edu* , poate trimite un pachet UDP, conținând numele gazdă, către un server DNS. Serverul răspunde cu un pachet UDP conținând adresa de IP a gazdei. Nu este necesară nici o inițializare în avans și nici o închidere de sesiune. Doar două mesaje traversează rețeaua.

6.4.2 Apel de procedură la distanță (Remote Procedure Call)

Într-un anume sens, trimiterea unui mesaj către o stație la distanță și primirea înapoi a unui răspuns seamănă mult cu realizarea unei funcții de apel într-un limbaj de programare. În ambele cazuri se începe cu unul sau mai mulți parametri și se primește înapoi un rezultat. Această observație le-a făcut pe unele persoane să încerce să organizeze interacțiunile cerere-răspuns în rețele pentru a fi puse împreună sub forma apelurilor procedurale. Un astfel de aranjament face aplicațiile de rețea mai ușor programabile și mai abordabile. De exemplu, imaginați-vă doar procedura numită *get_IP_address(host_name)* care funcționează prin trimiterea unui pachet UDP către un server DNS și așteptarea răspunsului, cronometrând și încercând încă o dată, dacă răspunsul nu apare suficient de rapid. În acest fel, toate detaliile de rețea pot fi ascunse programatorului.

Efortul cel mai important în acest domeniu a fost depus de către Birell și Nelson (1984). Rezumând, ce au sugerat Birell și Nelson a fost să permită programelor să apeleze proceduri localizate pe stații aflate la distanță. Când procesul de pe mașina 1 invocă o procedură de pe mașina 2, procesul apelant de pe prima mașină este suspendat și execuția procedurii invocate are loc pe cea de-a doua. Informația poate fi transportată de la cel care apelează la cel care este apelat în parametri și se poate întoarce în rezultatul procedurii. Nici un transfer de mesaje nu este vizibil pentru programator. Tehnica este cunoscută sub numele de **RPC** (**Remote Procedure Call** , rom: Apel de procedură la distanță), și a devenit baza pentru multe aplicații de rețea. În mod tradițional, procedura care apelează este cunoscută ca fiind clientul și procedura apelată ca fiind serverul și vom folosi aceste denumiri și aici.

Ideea din spatele RPC-ului este aceea de a face un apel de procedură la distanță să arate pe cât posibil ca unul local. În forma cea mai simplă, pentru apelarea unei proceduri la distanță, programul client trebuie sa fie legat cu o mică procedură de bibliotecă, numită **client stub** (rom: ciot), care reprezintă procedura server-ului în spațiul de adresă al clientului. În mod similar, serverul este legat cu

o procedură numită **server stub** . Aceste proceduri ascund faptul că apelul de procedură de la client la server nu este local.

Pașii efectivi ai realizării unui RPC sunt prezentați în fig. 6-24. Pasul 1 este cel în care clientul apelează stub-ul client. Acest apel este un apel de procedură locală, cu parametrii introduși în stivă în modul obișnuit. Pasul 2 constă în împachetarea parametrilor de către stub-ul client într-un mesaj și realizarea unui apel de sistem pentru a trimite mesajul. Împachetarea parametrilor este denumită **marshaling** (rom: împachetare). Pasul 3 constă în faptul că nucleul sistemului de operare trimite un mesaj de la mașina client la mașina server. Pasul 4 constă în trimiterea de către nucleu a pachetelor care sosesc la stub-ul server. În sfârșit, pasul 5 constă în faptul că stub-ul server apelează procedura server cu parametrii despachetați. Răspunsul urmează aceeași cale și în cealaltă direcție.

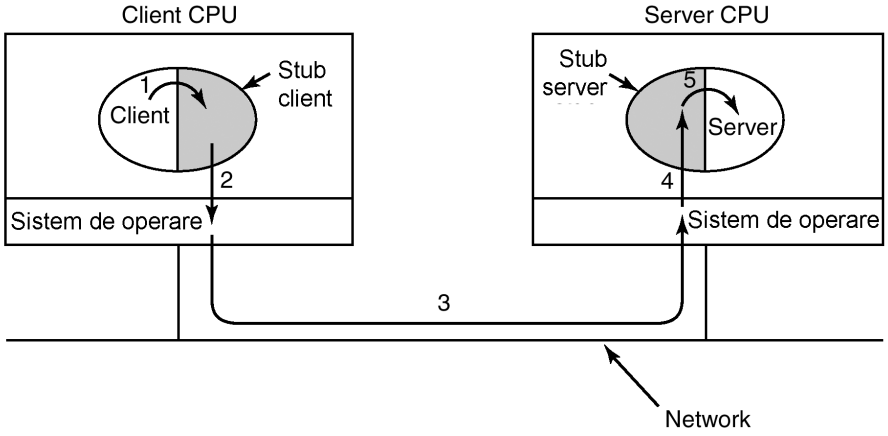


Fig. 6-24. Pașii pentru a crea un apel de procedură la distanță. Stub-urile sunt hașurate.

Elementul cheie de reținut aici este acela că procedura client, scrisă de către utilizator, doar face un apel normal de procedură (adică local) către stub-ul client, care are același nume cu procedura server. Cum procedura client și stub-ul client se găsesc în același spațiu de adresă, parametrii sunt transferați în modul obișnuit. În mod asemănător, procedura server este apelată de o procedură din spațiul său de adresă cu parametrii pe care îi așteaptă. Pentru procedura server nimic nu este neobișnuit. În felul acesta, în loc ca intrarea/ieșirea să se facă pe socluri, comunicația de rețea este realizată imitând o procedură de apelare normală.

În ciuda eleganței conceptului de RPC, „sunt câțiva șerpi care se ascund prin iarbă”. Unul mare este utilizarea parametrilor pointer. În mod normal, transmiterea unui pointer către procedură nu este o problemă. Procedura apelată poate folosi pointer-ul în același mod în care poate să o facă și cel care o apelează, deoarece ambele proceduri se găsesc în același spațiu de adrese virtuale. Cu RPC-ul, transmiterea pointer-ilor este imposibilă deoarece clientul și server-ul se găsesc în spații de adresă diferite.

În anumite cazuri, se pot folosi unele trucuri pentru a face posibilă transmiterea pointer-ilor. Să presupunem că primul parametru este un pointer către un întreg, *k*. Stub-ul client poate împacheta variabila *k* și să o trimită server-ului. Atunci, server stub creează un pointer către *k* și-l transmite procedurii server, exact așa cum aceasta se așteaptă. Când procedura server cedează controlul server stub, acesta din urmă trimite variabila *k* înapoi clientului, unde noul *k* este copiat peste cel vechi, în caz că serverul l-a schimbat. În fapt, secvența standard de apelare apel-prin-referință a fost înlocuită

de copiază-restaurează (eng.: copy-restore). Din păcate, acest truc nu funcționează întotdeauna, de exemplu dacă un pointer este către un grafic sau altă structură complexă de date. Din acest motiv, trebuie puse anumite restricții asupra parametrilor procedurilor apelate la distanță.

O a doua problemă este aceea că în limbajelor mai puțin bazate pe tipuri, cum ar fi C-ul, este perfect legal să scrii o procedură care calculează produsul scalar a doi vectori, fără a specifica dimensiunea vreunuia dintre ei. Fiecare poate fi terminat printr-o valoare specială cunoscută doar de către procedura apelată și de cea apelantă. În aceste condiții, în mod cert este imposibil pentru stub-ul client să împacheteze parametrii: nu are nici o modalitate de a determina cât de mult spațiu ocupă aceștia.

O a treia problemă este aceea că nu întotdeauna este posibilă deducerea tipurilor parametrilor, nici măcar dintr-o specificație formală sau din cod în sine. Un exemplu este *printf,* care poate avea orice număr de parametri (cel puțin unul), iar parametrii pot fi o combinație arbitrară a tipurilor întregi, short, long, caractere, șiruri, numere în virgulă mobilă de diferite lungimi și alte tipuri. A încerca să invoci *printf* ca procedură cu apel la distanță ar fi practic imposibil, deoarece C-ul este prea permisiv. Totuși, o regulă care să spună că RPC-ul poate fi folosit cu condiția să nu programezi în C (sau C++) nu ar fi prea populară.

O a patra problemă este legată de utilizarea variabilelor globale. În mod normal, procedura de apelare și cea care este apelată pot comunica folosind variabilele globale, în plus față de comunicarea prin parametri. Dacă procedura apelată este mutată acum pe o mașină la distanță, codul va da erori deoarece variabilele globale nu mai sunt partajate.

Aceste probleme nu sunt menite să sugereze că RPC-ul este lipsit de șanse. De fapt, este larg folosit, dar sunt necesare anumite restricții pentru a-l face să funcționeze bine în practică.

Desigur, RPC-ul nu are nevoie să folosească pachete UDP, dar RPC și UDP se potrivesc bine, și UDP este uzual folosit pentru RPC. Totuși, când parametrii sau rezultatele pot să fie mai mari decât pachetul maxim UDP sau atunci când operația cerută nu este idempotentă (adică nu poate fi repetată în siguranță, ca de exemplu atunci când se incrementează un contor), poate fi necesară stabilirea unei conexiuni TCP și trimiterea cererii prin aceasta, în loc să se folosească UDP-ul.

6.4.3 Protocolul de transport în timp real – Real-Time Transport Protocol

RPC-ul client-server este un domeniu în care UDP este mult folosit. Un alt domeniu este acela al aplicațiilor multimedia în timp real. În particular, având în vedere că radioul pe internet, telefonia pe Internet, muzica la cerere, video-conferințele, video la cerere și alte aplicații multimedia au devenit mai răspândite, oamenii au descoperit că fiecare aplicație a folosit, mai mult sau mai puțin, același protocol de transport în timp real. Treptat a devenit clar faptul că un protocol generic de transport în timp real, pentru aplicații multimedia, ar fi o idee bună. Așa a luat naștere **RTP**-ul (**Real-time Transport Protocol**, rom: Protocol de transport în timp real). Este descris în RFC 1889 și acum este folosit pe scară largă.

Poziția RTP-ului în stiva de protocoale este oarecum ciudată. S-a hotărât să se pună RTP-ul în spațiul utilizator și să se ruleze (în mod normal) peste UDP. El funcționează după cum urmează. Aplicațiile multimedia constau în aplicații audio, video, text și posibil alte fluxuri. Acestea sunt trimise bibliotecii RTP, care se află în spațiul utilizator împreună cu aplicația. Apoi, această bibliotecă multiplexează fluxurile și le codează în pachete RTP, pe care apoi le trimite printr-un soclu. La celălalt capăt al soclului (în nucleul sistemului de operare), pachete UDP sunt generate și încapsulate în pachete IP. Dacă computer-ul se găsește într-o rețea Ethernet, pachetele IP sunt puse apoi în cadre

Ethernet, pentru transmisie. Stiva de protocoale pentru această situație este prezentată în fig. 6-25 (a). Încapsularea pachetului este prezentată în fig. 6-25 (b).

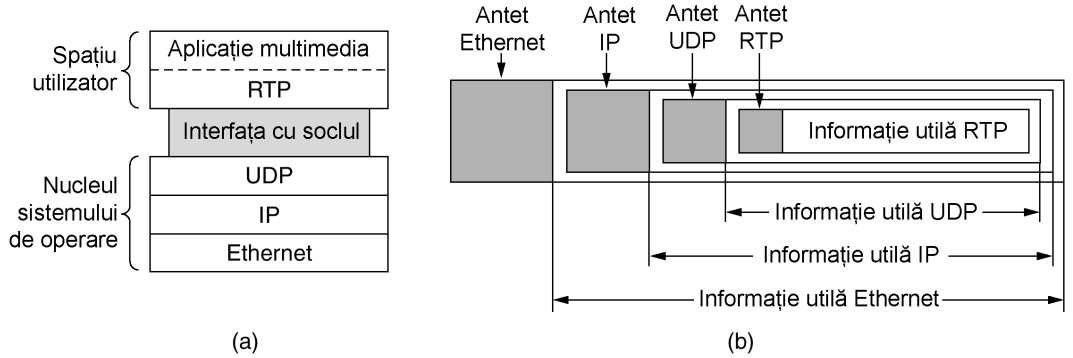


Fig. 6-25. (a)Poziționarea Protocolului RTP în stiva de protocoale. (b) Încapsularea pachetului.

Ca o consecință a acestei proiectări, este cam dificil de spus în ce nivel este RTP-ul. Cum rulează în spațiul utilizator și este legat la programul aplicație, în mod cert arată ca un protocol de aplicație. Pe de altă parte, este un protocol generic independent de aplicație, care doar furnizează facilități de transport, astfel încât arată totodată ca un protocol de transport. Probabil că cea mai potrivită descriere este aceea că este un protocol de transport care este implementat la nivelul aplicație.

Funcția de bază a RTP-ului este multiplexarea mai multor fluxuri de date în timp real într-un singur flux de pachete UDP. Fluxul UDP poate fi transmis către o singură destinație (unicasting) sau către destinații multiple (multicasting). Deoarece RTP-ul folosește numai UDP normal, pachetele sale nu sunt tratate în mod special de către rutere, decât dacă sunt activate anumite facilități de calitate a serviciilor din IP. În particular, nu există garanții speciale referitoare la livrare, bruiaj etc.

Fiecărui pachet trimis în fluxul RTP i se dă un număr cu unu mai mare decât al predecesorului său. Această numerotare permite destinației să stabilească dacă lipsesc unele pachete. Dacă un pachet lipsește, cea mai bună decizie ce poate fi luată de către destinație este de a aproxima valoarea lipsă prin interpolare. Retransmiterea nu este o opțiune practică având în vedere că pachetul retransmis va ajunge probabil prea târziu pentru a fi util. Ca o consecință, RTP-ul nu are control al fluxului, control al erorii, nu are confirmări și nu are mecanism pentru a cere retransmiterea.

Fiecare informație utilă din RTP poate să conțină mostre multiple și ele pot fi codate în orice mod dorește aplicația. Pentru a permite compatibilitatea , RTP-ul definește mai multe profiluri (de exemplu un singur flux audio) și pentru fiecare profil pot fi permise multiple formate de codare. De exemplu, un singur flux audio poate fi codat ca mostre de 8 biți PCM la 8 KHz, codare delta, codare previzibilă, codare GSM, MP3 și așa mai departe. RTP-ul furnizează un câmp antet în care sursa poate specifica codarea, dar altfel nu este implicat în modul în care este făcută codarea.

O altă facilitate de care au nevoie multe aplicații multimedia este stabilirea amprentei de timp. Aici ideea este de a permite sursei să asocieze o amprentă de timp cu prima mostră din fiecare pachet. Amprentele de timp sunt relative la începutul fluxului, așa că numai diferențele dintre acestea sunt semnificative. Valorile absolute nu au nici o semnificație. Acest mecanism permite destinației să folosească zone tampon de dimensiuni mici și să reproducă fiecare eșantion la numărul corect de milisecunde după începutul fluxului, independent de momentul în care ajunge pachetul ce conține eșantionul. Stabilirea amprentelor de timp nu numai că reduce efectele bruiajului, ci permite de asemenea mai multor fluxuri să se sincronizeze între ele. De exemplu, un program de televiziune

digital poate avea un flux video și două fluxuri audio. Cele două fluxuri audio pot fi pentru emisiuni stereo sau pentru a permite filmelor să fie manipulate cu o coloană sonoră în limba originală și o coloană sonoră dublată în limba locală, oferind o alegere celui care vede. Fiecare flux provine dintr-un dispozitiv fizic diferit, dar dacă sunt stabilite amprentele de timp de către un singur contor, ele pot fi redate sincronizat, chiar dacă fluxurile sunt transmise într-un mod dezordonat.

Antetul RTP este ilustrat în fig. 6-26. Acesta constă din trei cuvinte de 32 biți și eventual unele extensii. Primul cuvânt conține câmpul *Versiune*, care este deja la 2. Să sperăm că această versiune este foarte asemănătoare cu ultima versiune deoarece a mai rămas aici doar un punct de cod (deși 3 ar putea fi definit ca semnificând faptul că versiunea reală se găsește într-un cuvânt extins).

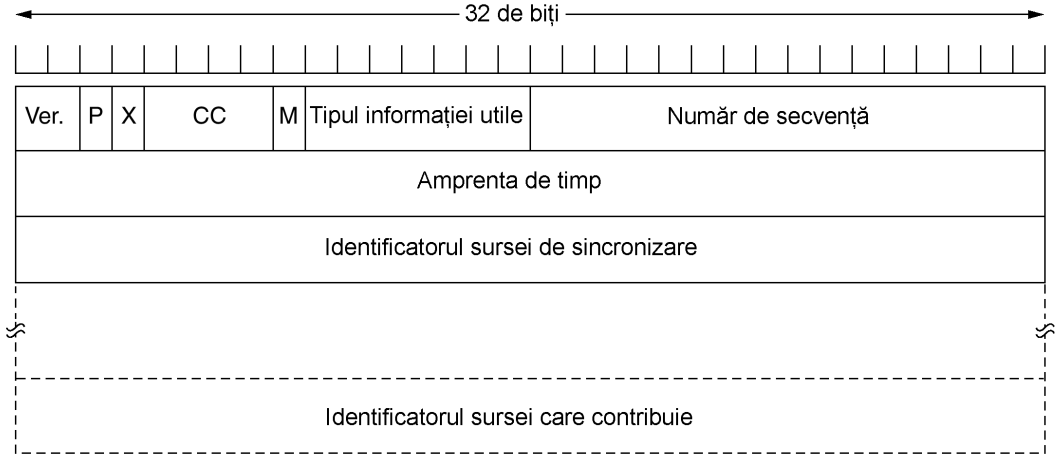


Fig. 6-26. Antetul RTP-ului.

Bitul *P* indică faptul că pachetul a fost extins la un multiplu de 4 octeți. Ultimul octet extins ne spune câți octeți au fost adăugați. Bitul *X* indică prezența unui antet extins. Formatul și semnificația antetului extins nu sunt definite. Singurul lucru care este definit este acela că primul cuvânt al extensiei dă lungimea. Aceasta este o cale de scăpare pentru orice cerințe neprevăzute.

Câmpul *CC* arată câte surse contribuabile sunt prezente, de la 0 la 15 (vezi mai jos). Bitul *M* este un bit de marcare specific aplicației. Poate fi folosit pentru a marca începutul unui cadru video, începutul unui cuvânt într-un canal audio sau altceva ce aplicația înțelege. Câmpul *Tip informație utilă* indică ce algoritm de codare a fost folosit (de exemplu 8 biți audio necompresați, MP3, etc). Din moment ce fiecare pachet transportă acest câmp, codarea se poate schimba în timpul transmisiei. *Numărul de secvență* este doar un contor care este incrementat pe fiecare pachet RTP trimis. Este folosit pentru a detecta pachetele pierdute.

Amprenta de timp este stabilită de către sursa fluxului pentru a se ști când este făcut primul eșantion din pachet. Această valoare poate să ajute la reducerea bruiajului la receptor prin separarea redării de momentul ajungerii pachetului. *Identificatorul sursei de sincronizare* spune cărui flux îi aparține pachetul. Este metoda utilizată pentru a multiplexa și demultiplexa mai multe fluxuri de date într-un singur flux de pachete UDP. În sfârșit, dacă există, *identificatorii sursei care contribuie* sunt folosiți când în studio există mixere. În acest caz, mixerul este sursa de sincronizare și fluxurile, fiind amestecate, apar aici.

RTP are un protocol înrudit numit **RTCP** (**Real Time Transport Control Protocol**, rom: Protocol de control al transportului în timp real). Acesta se ocupă de răspuns, sincronizare și de interfața cu utilizatorul, dar nu transportă date. Prima funcție poate fi folosită pentru a oferi surselor reacție (eng.: feedback) la întârzieri, bruiaj, lățime de bandă, congestie și alte proprietăți ale rețelei. Această informație poate fi folosită de către procesul de codare pentru a crește rata de transfer a datelor (și să ofere o calitate mai bună) când rețeaua merge bine și să reducă rata de transfer când apar probleme pe rețea. Prin furnizarea continuă de răspunsuri, algoritmii de codare pot fi în continuu adaptați pentru a oferi cea mai bună calitate posibilă în circumstanțele curente. De exemplu, dacă lățimea de bandă crește sau scade în timpul transmisiei, codarea poate să se schimbe de la MP3 la PCM pe 8 biți la codare delta, așa cum se cere. Câmpul *Tip informație utilă* este folosit pentru a spune destinației ce algoritm de codare este folosit pentru pachetul curent, făcând posibilă schimbarea acestuia la cerere.

De asemenea, RTCP-ul se ocupă de sincronizarea între fluxuri. Problema este că fluxuri diferite pot folosi ceasuri diferite, cu granularități diferite și devieri de flux diferite. RTCP poate fi folosit pentru a le menține sincronizate.

În sfârșit, RTCP oferă un mod pentru a numi diversele surse (de exemplu în text ASCII). Această informație poate fi afișată pe ecranul receptorului pentru a indica cine vorbește în acel moment.

Mai multe informații despre RTP pot fi găsite în (Perkins, 2002).

6.5. PROTOCOALE DE TRANSPORT PRIN INTERNET: TCP

UDP-ul este un protocol simplu și are anumite nișe de utilizare, cum ar fi interacțiunile client-server și cele multimedia, dar pentru cele mai multe aplicații de Internet este necesar un transport de încredere, secvențial al informației. UDP-ul nu poate oferi acest lucru, deci este nevoie de un alt protocol. Acesta este TCP și este pionul principal de lucru al Internet-ului. Să-l studiem acum în amănunt.

6.5.1 Introducere în TCP

TCP (**Transport Communication Protocol** - protocol de comunicație de nivel transport) a fost proiectat explicit pentru a asigura un flux sigur de octeți de la un capăt la celălalt al conexiunii într-o inter-rețea nesigură. O inter-rețea diferă de o rețea propriu-zisă prin faptul că diferite părți ale sale pot diferi substanțial în topologie, lărgime de bandă, întârzieri, dimensiunea pachetelor și alți parametri. TCP a fost proiectat să se adapteze în mod dinamic la proprietățile inter-rețelei și să fie robust în ceea ce privește mai multe tipuri de defecte.

TCP a fost definit în mod oficial în RFC 793. O dată cu trecerea timpului, au fost detectate diverse erori și inconsistențe și au fost modificate cerințele în anumite subdomenii. Aceste clarificări, precum și corectarea câtorva erori sunt detaliate în RFC 1122. Extensiile sunt furnizate în RFC 1323.

Fiecare mașină care suportă TCP dispune de o entitate de transport TCP, fie ca proces utilizator, fie ca procedură de bibliotecă, fie ca parte a nucleului. În toate aceste cazuri, ea care gestionează fluxurile TCP și interfețele către nivelul IP. O entitate TCP acceptă fluxuri de date utilizator de la procesele locale, le împarte în fragmente care nu depășesc 64K octeți (de regulă în fragmente de aproximativ 1460 de octeți, pentru a încăpea într-un singur cadru Ethernet împreună cu antetele

TCP și IP) și expediază fiecare fragment ca o datagramă IP separată. Atunci când datagramele IP conținând informație TCP sosesc la o mașină, ele sunt furnizate entității TCP, care reconstruiește fluxul original de octeți. Pentru simplificare, vom folosi câteodată doar TCP , subînțelegând prin aceasta sau entitatea TCP de transport (o porțiune de program) sau protocolul TCP (un set de reguli). Din context va fi clar care din cele două noțiuni este referită. De exemplu, în „Utilizatorul furnizează date TCP-ului" este clară referirea la entitatea TCP de transport.

Nivelul IP nu oferă nici o garanție că datagramele vor fi livrate corect, astfel că este sarcina TCP-ului să detecteze eroarea și să efectueze o retransmisie atunci când situația o impune. Datagramele care ajung (totuși) la destinație pot sosi într-o ordine eronată; este, de asemenea, sarcina TCP-ului să le reasambleze în mesaje respectând ordinea corectă (de secvență). Pe scurt, TCP-ul trebuie să ofere fiabilitatea pe care cei mai mulți utilizatori o doresc și pe care IP-ul nu o oferă.

6.5.2 Modelul serviciului TCP

Serviciul TCP este obținut prin crearea atât de către emițător, cât și de către receptor, a unor puncte finale, numite socluri (sockets), așa cum s-a discutat în Sec. 6.1.3. Fiecare soclu are un număr de soclu (adresă) format din adresa IP a mașinii gazdă și un număr de 16 biți, local gazdei respective, numit **port**. Port este numele TCP pentru un TSAP. Pentru a obține o conexiune TCP, trebuie stabilită explicit o conexiune între un soclu de pe mașina emițătoare și un soclu de pe mașina receptoare. Apelurile de soclu sunt prezentate în fig. 6-5.

Un soclu poate fi folosit la un moment dat pentru mai multe conexiuni. Altfel spus, două sau mai multe conexiuni se pot termina la același soclu. Conexiunile sunt identificate prin identificatorii soclurilor de la ambele capete, adică *(soclu 1, soclu 2)*. Nu este folosit nici un alt număr sau identificator de circuit virtual.

Numerele de port mai mici decât 256 se numesc **porturi general cunoscute** și sunt rezervate serviciilor standard. De exemplu, orice proces care dorește să stabilească o conexiune cu o mașină gazdă pentru a transfera un fișier utilizând FTP, se poate conecta la portul 21 al mașinii destinație pentru a contacta demonul său FTP. Similar, portul 23 este folosit pentru a stabili o sesiune de lucru la distanță utilizând TELNET. Lista porturilor general cunoscute se găsește la *www.iana.org*. Câteva dintre cele foarte cunoscute sunt prezentate în fig. 6-27.

Port	Protocol	Utilitate
21	FTP	Transfer de fișiere
23	Telnet	Login la distanță
25	SMTP	E-mail
69	TFTP	Protocol de transfer de fișiere trivial
79	Finger	Căutare de informații despre un utilizator
80	HTTP	World Wide Web
110	POP-3	Acces prin e-mail la distanță
119	NNTP	Știri USENET

Fig. 6-27. Câteva porturi asignate.

Cu siguranță ar fi posibil ca, în momentul încărcării, demonul de FTP să se autoatașeze la portul 21, demonul telnet la portul 23 și tot așa. Totuși, dacă s-ar proceda astfel s-ar umple memoria cu demoni inactivi în majoritatea timpului. În schimb, în general se folosește un singur demon, numit **inetd** (**Internet daemon**, rom: demon de Internet) în UNIX, care să se autoatașeze la mai multe porturi și să aștepte prima conexiune care vine. Când acest lucru se întâmplă, inetd creează un nou pro-

ces și execută în el demonul adecvat, lăsând acel demon să se ocupe de cerere. Astfel, demonii, în afară de inetd, sunt activi doar când au de lucru. Inetd află ce porturi să folosească dintr-un fișier de configurare. În consecință, administratorul de sistem poate seta sistemul să aibă demoni permanenți pe cele mai ocupate porturi (de exemplu portul 80) și inetd pe restul.

Toate conexiunile TCP sunt duplex integral și punct-la-punct. Duplex integral înseamnă că traficul se poate desfășura în ambele sensuri în același timp. Punct-la-punct indică faptul că fiecare conexiune are exact două puncte finale. TCP nu suportă difuzarea parțială sau totală.

O conexiune TCP este un flux de octeți și nu un flux de mesaje. Dimensiunile mesajelor nu se conservă de la un capăt la celălalt. De exemplu, dacă procesul emițător execută patru scrieri de câte 512 octeți pe un canal TCP, aceste date pot fi livrate procesului receptor ca patru fragmente (chunks) de 512 octeți, două fragmente de 1024 octeți, un singur fragment de 2048 octeți (vezi fig. 6-28) sau în orice alt mod. Nu există posibilitatea ca receptorul să determine numărul de unități în care a fost scrisă informația.

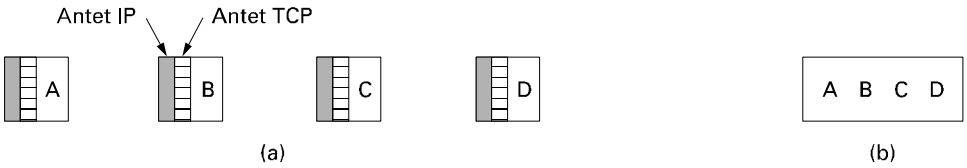


Fig. 6-28. (a) Patru segmente de 512 octeți au fost trimise ca datagrame IP separate. (b) Livrarea celor 2048 octeți către aplicație, printr-un singur apel *read*.

În UNIX, aceeași proprietate o au și fișierele. Cititorul unui fișier nu poate spune dacă fișierul a fost scris bloc cu bloc, octet cu octet sau tot dintr-o dată. Ca și un fișier UNIX, programele TCP nu au nici cea mai vagă idee despre semnificația octeților și nici cel mai mic interes pentru a afla acest lucru. Un octet este pur și simplu un octet.

Atunci când o aplicație trimite date către TCP, TCP-ul le poate expedia imediat sau le poate reține într-un tampon (în scopul colectării unei cantități mai mari de informație pe care să o expedieze toată odată), după bunul său plac. Cu toate acestea, câteodată, aplicația dorește ca informația să fie expediată imediat. De exemplu, să presupunem că un utilizator este conectat la o mașină de la distanță. După ce a fost terminată o linie de comandă și s-a tastat Return, este esențial ca linia să fie imediat expediată către mașina de la distanță și să nu fie memorată până la terminarea următoarei linii. Pentru a forța expedierea, aplicația poate folosi indicatorul PUSH, care îi semnalează TCP-ului să nu întârzie procesul de transmisie.

Unele din primele aplicații foloseau indicatorul PUSH ca un fel de marcaj pentru a delimita marginile mesajelor. Deși acest truc funcționează câteodată, uneori el eșuează datorită faptului că, la recepție, nu toate implementările TCP-ului transmit aplicației indicatorul PUSH. Mai mult decât atât, dacă mai multe indicatoare PUSH apar înainte ca primul să fi fost transmis (de exemplu, pentru că linia de legătură este ocupată), TCP-ul este liber să colecteze toată informația referită de către aceste indicatoare într-o singură datagramă IP, fără să includă nici un separator între diferitele sale părți.

O ultimă caracteristică a serviciului TCP care merită menționată aici constă în **informația urgentă**. Atunci când un utilizator apasă tasta DEL sau CTRL-C pentru a întrerupe o prelucrare la distanță, aflată deja în execuție, aplicația emițător plasează o informație de control în fluxul de date și o furnizează TCP-ului împreună cu indicatorul URGENT. Acest eveniment impune TCP-ului întreruperea acumulării de informație și transmisia imediată a întregii informații disponibile deja pentru conexiunea respectivă.

Atunci când informația urgentă este recepționată la destinație, aplicația receptoare este întreruptă (de ex. prin emisia unui semnal, în terminologie UNIX), astfel încât, eliberată de orice altă activitate, aplicația să poată citi fluxul de date și să poată regăsi informația urgentă. Sfârșitul informației urgente este marcat, astfel încât aplicația să știe când se termină informația. Începutul informației urgente nu este marcat. Este sarcina aplicației să determine acest început. Această schemă furnizează de fapt un rudiment de mecanism de semnalizare, orice alte detalii fiind lăsate la latitudinea aplicației.

6.5.3 Protocolul TCP

În această secțiune vom prezenta un punct de vedere general asupra protocolului TCP, pentru a ne concentra apoi, în secțiunea care îi urmează, asupra antetului protocolului, câmp cu câmp.

O caracteristică importantă a TCP, care domină structura protocolului, este aceea că fiecare octet al unei conexiuni TCP are propriul său număr de secvență, reprezentat pe 32 biți. Când a luat ființă Internetul, liniile dintre rutere erau în cel mai bun caz linii închiriate de 56 Kbps, deci unei gazde funcționând la viteză maximă îi lua mai mult de o săptămână să utilizeze toate numerele de secvență. La vitezele rețelelor moderne, numerele de secvență pot fi consumate intr-un ritm alarmant, după cum vom vedea mai târziu. Numerele de secvență sunt utilizate atât pentru confirmări cât și pentru mecanismul de secvențiere, acesta din urmă utilizând câmpuri separate de 32 de biți din antet.

Entitățile TCP de transmisie și de recepție interschimbă informație sub formă de segmente. Un **segment TCP** constă dintr-un antet de exact 20 de octeți (plus o parte opțională) urmat de zero sau mai mulți octeți de date. Programul TCP este cel care decide cât de mari trebuie să fie aceste segmente. El poate acumula informație provenită din mai multe scrieri într-un singur segment sau poate fragmenta informația provenind dintr-o singură scriere în mai multe segmente. Există două limite care restricționează dimensiunea unui segment. În primul rând, fiecare segment, inclusiv antetul TCP, trebuie să încapă în cei 65.535 de octeți de informație utilă IP. În al doilea rând, fiecare rețea are o **unitate maximă de transfer** sau **MTU (Maximum Transfer Unit)**, deci fiecare segment trebuie să încapă în acest MTU. În realitate, MTU este în general de 1500 octeți (dimensiunea informației utile din Ethernet), definind astfel o limită superioară a dimensiunii unui segment.

Protocolul de bază utilizat de către entitățile TCP este protocolul cu fereastră glisantă. Atunci când un emițător transmite un segment, el pornește un cronometru. Atunci când un segment ajunge la destinație, entitatea TCP receptoare trimite înapoi un segment (cu informație utilă, dacă aceasta există sau fără, în caz contrar) care conține totodată și numărul de secvență următor pe care aceasta se așteaptă să-l recepționeze. Dacă cronometrul emițătorului depășește o anumită valoare înaintea primirii confirmării, emițătorul retransmite segmentul neconfirmat.

Deși acest protocol pare simplu, pot apărea multe situații particulare pe care le vom prezenta mai jos. Segmentele pot ajunge într-o ordine arbitrară, deci octeții 3072-4095 pot fi recepționați, dar nu pot fi confirmați datorită absenței octeților 2048-3071. Segmentele pot de asemenea întârzia pe drum un interval de timp suficient de mare pentru ca emițătorul să detecteze o depășire a cronometrului și să le retransmită. Retransmisiile pot include porțiuni de mesaj fragmentate altfel decât în transmisia inițială, ceea ce impune o tratare atentă, astfel încât să se țină evidența octeților primiți corect. Totuși, deoarece fiecare octet din flux are un deplasament unic față de începutul mesajului, acest lucru se poate realiza.

TCP trebuie să fie pregătit să facă față unor astfel de situații și să le rezolve într-o manieră eficientă. Un efort considerabil a fost dedicat optimizării performanțelor fluxurilor TCP, ținându-se cont inclusiv de probleme legate de rețea. În continuare vor fi prezentați un număr de algoritmi utilizați de numeroase implementări TCP.

6.5.4 Antetul segmentului TCP

În fig. 6-29 este prezentată structura unui segment TCP. Fiecare segment începe cu un antet format dintr-o structură fixă de 20 de octeți. Antetul fix poate fi urmat de un set de opțiuni asociate antetului. În continuarea opțiunilor, dacă ele există, pot urma până la 65.535 - 20 - 20 = 65.495 de octeți de date, unde primul 20 reprezintă antetul IP, iar al doilea antetul TCP. Segmente care nu conțin octeți de date sunt nu numai permise, dar și utilizate în mod frecvent pentru confirmări și mesaje de control.

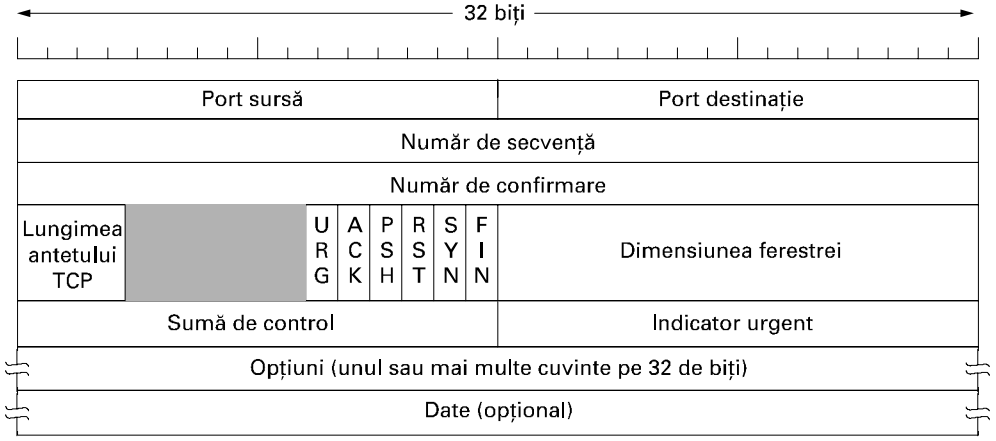


Fig. 6-29. Antetul TCP.

Să disecăm acum structura antetului TCP, câmp cu câmp. Câmpurile *Port sursă* și *Port destinație* identifică punctele finale ale conexiunii. Porturile general cunoscute sunt definite la *www.iana.org*, dar fiecare gazdă le poate aloca pe celelalte după cum dorește. Un port formează împreună cu adresa IP a mașinii sale un unic punct de capăt (eng.: end point) de 48 de biți. Conexiunea este identificată de punctele de capăt ale sursei și destinației.

Câmpurile *Număr de secvență* și *Număr de confirmare* au semnificația funcțiilor lor uzuale. Trebuie observat că cel din urmă indică octetul următor așteptat și nu ultimul octet recepționat în mod corect. Ambele câmpuri au lungimea de 32 de biți, deoarece într-un flux TCP fiecare bit de informație este numerotat.

Lungimea antetului TCP indică numărul de cuvinte de 32 de biți care sunt conținute în antetul TCP. Această informație este utilă, deoarece câmpul *Opțiuni* este de lungime variabilă, proprietate pe care o transmite astfel și antetului. Tehnic vorbind, acest câmp indică în realitate începutul datelor din segment, măsurat în cuvinte de 32 de biți, dar cum acest număr este identic cu lungimea antetului în cuvinte, efectul este același.

Urmează un câmp de șase biți care este neutilizat. Faptul că acest câmp a supraviețuit intact mai mult de un sfert de secol este o mărturie despre cât de bine a fost proiectat TCP-ul. Protocoale mai prost concepute ar fi avut nevoie de el pentru a corecta erori ale proiectării inițiale.

Urmează acum șase indicatori de câte un bit. URG este poziționat pe 1 dacă Indicatorul Urgent este valid. Indicatorul Urgent este folosit pentru a indica deplasamentul în octeți față de numărul curent de secvență la care se găsește informația urgentă. O astfel de facilitate ține locul mesajelor de

întrerupere. Așa cum am menționat deja anterior, această facilitate reprezintă esența modului în care emițătorul poate transmite un semnal receptorului fără ca TCP-ul în sine să fie cauza întreruperii.

Bitul *ACK* este poziționat pe 1 pentru a indica faptul că *Numărul de confirmare* este valid. În cazul în care *ACK* este poziționat pe 0, segmentul în discuție nu conține o confirmare și câmpul *Număr de confirmare* este ignorat.

Bitul *PSH* indică informația FORȚATĂ. Receptorul este rugat respectuos să livreze aplicației informația respectivă imediat ce este recepționată și să nu o memoreze în așteptarea umplerii tampoanelor de comunicație (lucru care, altminteri, ar fi făcut din rațiuni de eficiență).

Bitul *RST* este folosit pentru a desființa o conexiune care a devenit inutilizabilă datorită defecțiunii unei mașini sau oricărui alt motiv. El este de asemenea utilizat pentru a refuza un segment invalid sau o încercare de deschidere a unei conexiuni. În general, recepționarea unui segment având acest bit poziționat indică o problemă care trebuie tratată în funcție de context.

Bitul *SYN* este utilizat pentru stabilirea unei conexiuni. Cererea de conexiune conține $SYN = 1$ și $ACK = 0$ pentru a indica faptul că acel câmp suplimentar de confirmare nu este utilizat. Răspunsul la o astfel de cerere conține o confirmare, având deci $SYN = 1$ și $ACK = 1$. În esență, bitul *SYN* este utilizat pentru a indica o CERERE DE CONEXIUNE și o CONEXIUNE ACCEPTATĂ, bitul *ACK* făcând distincția între cele două posibilități.

Bitul *FIN* este folosit pentru a încheia o conexiune. El indică faptul că emițătorul nu mai are nici o informație de transmis. Cu toate acestea, după închiderea conexiunii, un proces poate recepționa în continuare date pe o durată nedefinită. Ambele segmente, *SYN* și *FIN,* conțin numere de secvență și astfel este garantat faptul că ele vor fi prelucrate în ordinea corectă.

În TCP, fluxul de control este tratat prin ferestre glisante de dimensiune variabilă. Câmpul *Fereastră* indică numărul de octeți care pot fi trimiși, începând de la octetul confirmat. Un câmp *Fereastră* de valoare 0 este perfect legal și spune că octeții până la *Număr de confirmare* - 1 inclusiv au fost recepționați, dar receptorul dorește cu ardoare o pauză, așa că mulțumește frumos, dar pentru moment nu dorește continuarea transferului. Permisiunea de expediere poate fi acordată ulterior de către receptor prin trimiterea unui segment având același *Număr de confirmare,* dar un câmp *Fereastră* cu o valoare nenulă.

În protocoalele din cap. 3, confirmările pentru cadrele primite și permisiunea de a trimite noi cadre erau legate una de alta. Aceasta era o consecință a dimensiunii fixe a ferestrei pentru fiecare protocol. În TCP, confirmările și permisiunea de a trimite noi date sunt total decuplate. De fapt, receptorul poate spune: Am primit octeții până la al *k*-lea, dar în acest moment nu mai doresc să primesc alții. Această decuplare (care de fapt reprezintă o fereastră de dimensiune variabilă) oferă mai multă flexibilitate. O vom studia detaliat mai jos.

Este de asemenea prevăzută o *Sumă de control,* în scopul obținerii unei fiabilități extreme. Această sumă de control este calculată pentru antet, informație și pseudo-antetul conceptual prezentat în fig. 6-30. În momentul calculului, *Suma de control* TCP este poziționată pe zero, iar câmpul de date este completat cu un octet suplimentar nul, dacă lungimea sa este un număr impar. Algoritmul de calcul al sumei de control este simplu, el adunând toate cuvintele de 16 biți în complement față de 1 și aplicând apoi încă o dată complementul față de 1 asupra sumei. În acest mod, atunci când receptorul aplică același calcul asupra întregului segment, inclusiv asupra *Sumei de control*, rezultatul ar trebui să fie 0.

Pseudo-antetul conține adresele IP ale mașinii sursă și destinație, de 32 de biți fiecare, numărul de protocol pentru TCP (6) și numărul de octeți al segmentului TCP (incluzând și antetul). Prin includerea pseudo-antetului în calculul sumei de control TCP se pot detecta pachetele care au fost

preluate eronat, dar procedând astfel, este negată însăși ierarhia protocolului, deoarece adresa IP aparține nivelului IP și nu nivelului TCP.

Câmpul *Opțiuni* a fost proiectat pentru a permite adăugarea unor facilități suplimentare neacoperite de antetul obișnuit. Cea mai importantă opțiune este aceea care permite fiecărei mașini să specifice încărcarea maximă de informație utilă TCP pe care este dispusă să o accepte. Utilizarea segmentelor de dimensiune mare este mai eficientă decât utilizarea segmentelor de dimensiune mică datorită amortizării antetului de 20 de octeți prin cantitatea mai mare de informație utilă. Cu toate acestea, este posibil ca mașini mai puțin performante să nu fie capabile să manevreze segmente foarte mari. În timpul inițializării conexiunii, fiecare parte anunță dimensiunea maximă acceptată și așteaptă de la partener aceeași informație. Câștigă cel mai mic dintre cele două numere. Dacă o mașină nu folosește această opțiune, cantitatea implicită de informație utilă este de 536 octeți. Toate mașinile din Internet trebuie să accepte segmente de dimensiune 536 + 20 = 556 octeți. Dimensiunea maximă a segmentului nu trebuie să fie aceeași în cele două direcții.

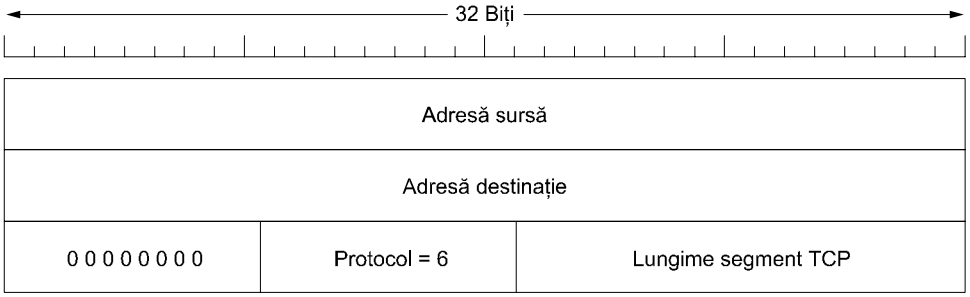


Fig. 6-30. Pseudo-antetul inclus în suma de control TCP.

O fereastră de 64 K octeți reprezintă adesea o problemă pentru liniile cu o lărgime de bandă mare și/sau cu întârzieri mari. Pe o linie T3 (44.736 Mbps) trimiterea unei ferestre întregi de 64 K octeți durează doar 12 ms. Dacă întârzierea propagării dus-întors este de 50 ms (care este valoarea tipică pentru o linie trans-continentală), emițătorul va aștepta confirmări - fiind deci inactiv ¾ din timp. Pe o conexiune prin satelit, situația este chiar mai rea. O fereastră de dimensiune mare ar permite emițătorului să continue trimiterea informației, însă o astfel de dimensiune nu poate fi reprezentată în cei 16 biți ai câmpului Fereastră. În RFC 1323 se propune o opțiune Scală a ferestrei, permițând emițătorului și receptorului să negocieze un factor de scalare a ferestrei. Acest număr permite ambelor părți să deplaseze câmpul Fereastră cu până la 14 biți spre stânga, permițând astfel ferestre de până la 230 octeți. Această opțiune este suportată în prezent de cele mai multe implementări ale TCP-ului.

O altă opțiune propusă de RFC 1106, și care este în prezent implementată pe scară largă, constă în utilizarea unei repetări selective în locul unui protocol cu întoarcere de n pași (eng.: go back n protocol). Dacă receptorul primește un segment eronat urmat de un număr mare de segmente corecte, protocolul TCP clasic va constata într-un final o depășire de timp și va retrimite toate segmentele neconfirmate, deci și pe acelea care au fost recepționate corect (adică se face o întoarcere de n pași). RFC 1106 introduce NAK-urile pentru a permite receptorului să ceară un anumit segment (sau segmente). După obținerea acestora, el poate confirma toată informația memorată reducând astfel cantitatea de informație retransmisă.

6.5.5 Stabilirea conexiunii TCP

În TCP conexiunile sunt stabilite utilizând „înțelegerea în trei pași", discutată în Sec. 6.2.2. Pentru a stabili o conexiune, una din părți - să spunem serverul - așteaptă în mod pasiv o cerere de conexiune prin execuția primitivelor LISTEN și ACCEPT, putând specifica o sursă anume sau nici o sursă în mod particular.

Cealaltă parte - să spunem clientul - execută o primitivă CONNECT, indicând adresa IP și numărul de port la care dorește să se conecteze, dimensiunea maximă a segmentului TCP pe care este dispusă să o accepte și, opțional, o informație utilizator (de exemplu o parolă). Primitiva CONNECT trimite un segment TCP având bitul *SYN* poziționat și bitul *ACK* nepoziționat, după care așteaptă un răspuns.

Atunci când sosește la destinație un segment, entitatea TCP receptoare verifică dacă nu cumva există un proces care a executat LISTEN pe numărul de port specificat în câmpul *Port destinație*. În caz contrar, trimite un răspuns cu bitul *RST* poziționat, pentru a refuza conexiunea.

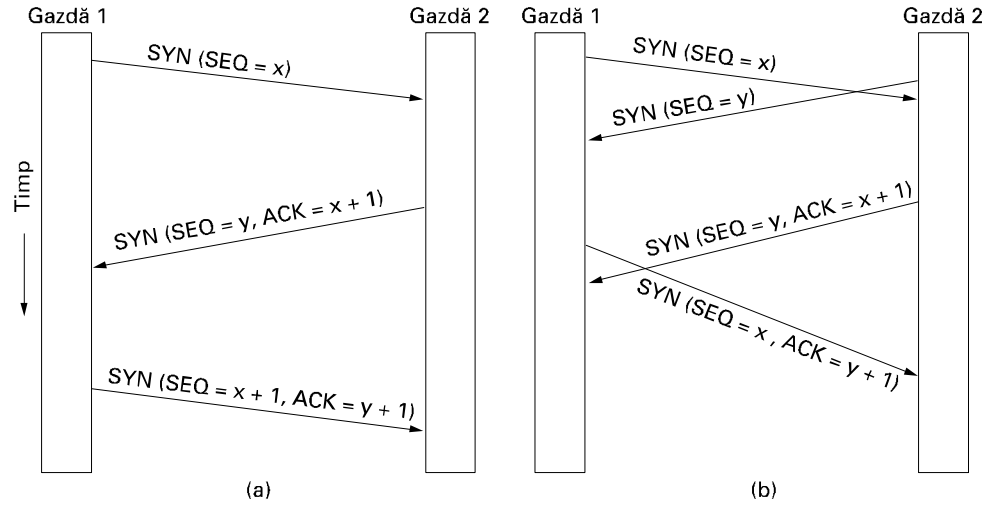

Gazdă 1
Gazdă 2
SYN (SEQ = x)
Timp
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x + 1, ACK = y + 1)
(a)
Gazdă 1
Gazdă 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x, ACK = y + 1)
(b)

Fig. 6-31. (a) Stabilirea unei conexiuni TCP în cazul normal. (b) Coliziunea apelurilor.

Dacă există vreun proces care ascultă la acel port, segmentul TCP recepționat va fi dirijat către procesul respectiv. Acesta poate accepta sau refuza conexiunea. Dacă o acceptă, trimite înapoi expeditorului un segment de confirmare. În fig. 6-31(a) este reprezentată secvența de segmente TCP transferate în caz de funcționare normală. De notat că un segment *SYN* consumă un octet din spațiul numerelor de secvență, astfel încât confirmarea să poată fi făcută fără ambiguități.

Secvența de evenimente ilustrată în fig. 6-31(b) reprezintă cazul în care două mașini încearcă simultan să stabilească o conexiune între aceleași două porturi. Rezultă că este suficient să fie stabilită o singură conexiune și nu două, deoarece conexiunile sunt identificate prin punctele lor terminale. Dacă prima inițializare conduce la crearea unei conexiuni identificată prin *(x, y)* și același lucru îl face și cea de-a doua inițializare, atunci este construită o singură intrare de tabel, în speță pentru *(x, y)*.

Numărul inițial de secvență asociat unei conexiuni nu este 0, din motivele discutate anterior. Se utilizează o schemă bazată pe un ceas cu o bătaie la fiecare 4 μs. Pentru mai multă siguranță , atunci când

o mașină se defectează, este posibil ca ea să nu fie reinițializată în timpul de viață maxim al unui pachet, garantându-se astfel că pachetele unei conexiuni anterioare nu se plimbă încă pe undeva prin Internet.

6.5.6 Eliberarea conexiunii TCP

Deși conexiunile TCP sunt bidirecționale, pentru a înțelege cum sunt desființate conexiunile, cel mai bine este să ni le imaginăm sub forma unei perechi de legături unidirecționale. Fiecare legătură unidirecțională este eliberată independent de perechea sa. Pentru eliberarea unei conexiuni, orice partener poate expedia un segment TCP având bitul *FIN* setat, lucru care indică faptul că nici o informație nu mai urmează să fie transmisă. Atunci când *FIN*-ul este confirmat, sensul respectiv de comunicare este efectiv oprit pentru noi date. Cu toate acestea, informația poate fi transferată în continuare, pentru un timp nedefinit, în celălalt sens. Conexiunea este desființată atunci când ambele direcții au fost oprite. În mod normal, pentru a elibera o conexiune sunt necesare patru segmente TCP: câte un *FIN* și un *ACK* pentru fiecare sens. Cu toate acestea, este posibil ca primul *ACK* și cel de-al doilea *FIN* să fie cuprinse în același segment reducând astfel numărul total la trei.

La fel ca în conversațiile telefonice, în care ambele persoane pot spune „la revedere” și pot închide telefonul simultan, ambele capete ale unei conexiuni TCP pot expedia segmente *FIN* în același timp. Acestea sunt confirmate ca de obicei, conexiunea fiind astfel eliberată. Nu există de fapt nici o diferență esențială între cazurile în care mașinile eliberează conexiunea secvențial respectiv simultan.

Pentru a evita problema celor două armate, sunt utilizate cronometre. Dacă un răspuns la un *FIN* nu este recepționat pe durata a cel mult două cicluri de maxime de viață ale unui pachet, emițătorul *FIN*-ului eliberează conexiunea. Cealaltă parte va observa în final că nimeni nu mai pare să asculte la celălalt capăt al conexiunii, și va elibera conexiunea în urma expirării unui interval de timp. Această soluție nu este perfectă, dar având în vedere faptul că o soluție perfectă este teoretic imposibilă, va trebui să ne mulțumim cu ce avem. În realitate astfel de probleme apar foarte rar.

6.5.7 Modelarea administrării conexiunii TCP

Pașii necesari stabilirii unei conexiuni pot fi reprezentați printr-un automat cu stări finite, cele 11 stări ale acestuia fiind prezentate în fig. 6-32. În fiecare stare pot apărea doar anumite evenimente. Atunci când are loc un astfel de eveniment, este îndeplinită o acțiune specifică. Atunci când se produce un eveniment a cărui apariție nu este legală în starea curentă, este semnalată o eroare.

Stare	Descriere
CLOSED (ÎNCHIS)	Nici o conexiune nu este activă sau în așteptare
LISTEN (ASCULTARE)	Serverul așteaptă recepționarea unui apel
SYN RCVD (Recepție SYN)	S-a recepționat o cerere de conexiune; aștept ACK
SYN SENT (Transmisie SYN)	Aplicația a început deschiderea unei conexiuni
ESTABLISHED (STABILIT)	Starea normală de transfer a datelor
FIN WAIT 1 (Așteptare FIN 1)	Aplicația a anunțat că termină
FIN WAIT 2 (Așteptare FIN 2)	Partenerul este de acord cu eliberarea conexiunii
TIMED WAIT (Așteptare Temporizată)	Se așteaptă „moartea” tuturor pachetelor
CLOSING (În curs de INCHIDERE)	Ambele părți încearcă simultan închiderea
CLOSE WAIT (ÎNCHIDE și AȘTEAPTĂ)	Partenerul a inițiat eliberarea conexiunii
LAST ACK (CONFIRMARE FINALĂ)	Se așteaptă „moartea” tuturor pachetelor

Fig. 6-32. Stările utilizate în automatul cu stări finite pentru controlul conexiunii TCP.

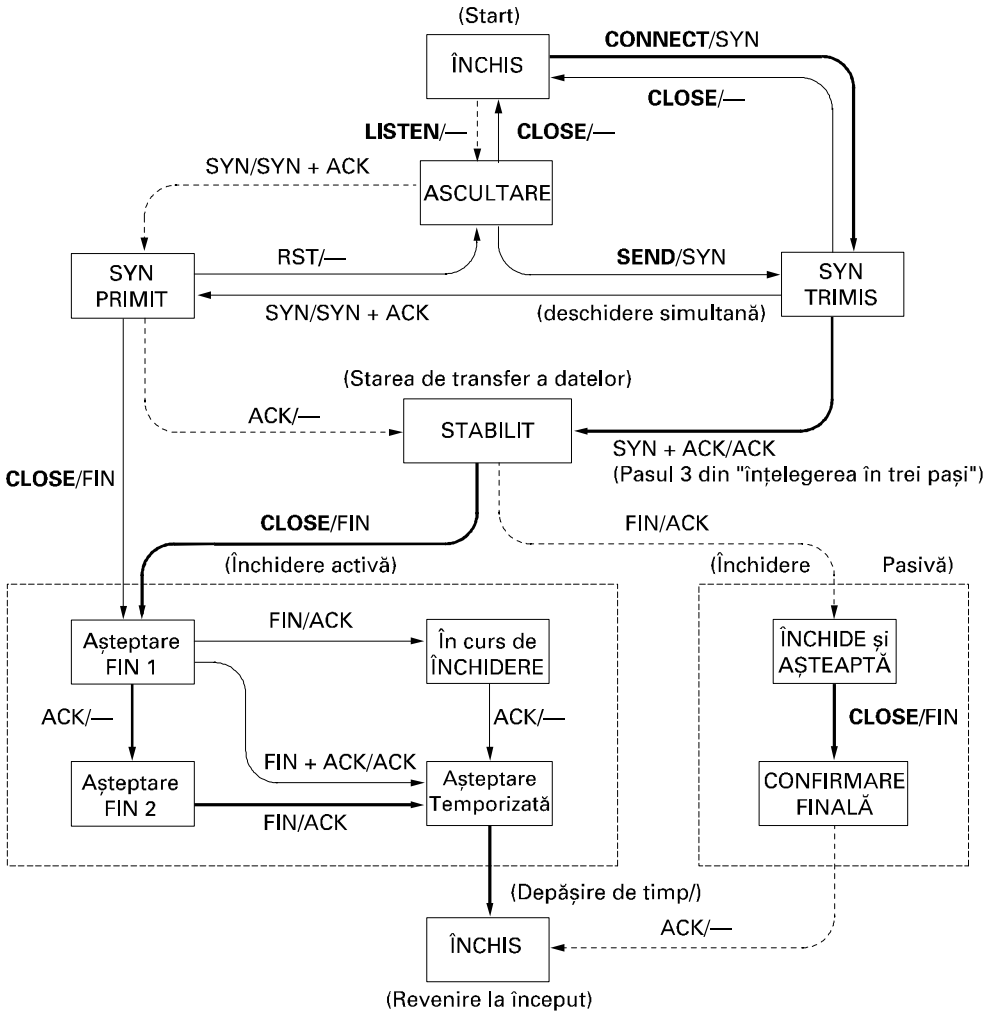


Fig. 6-33. Automatul cu stări finite pentru controlul conexiunii TCP. Linia groasă continuă este calea normală pentru client. Linia groasă întreruptă este calea normală pentru server. Liniile subțiri sunt evenimente neuzuale. Fiecare tranziție este etichetată de evenimentul care a creat-o și acțiunea care rezultă din el, separate de slash.

Fiecare conexiune începe în starea *ÎNCHIS*. Această stare este părăsită dacă urmează să se stabilească o conexiune pasivă (LISTEN) sau activă (CONNECT). Dacă partenerul stabilește o conexiune de tipul opus, starea devine *STABILIT*. Desființarea conexiunii poate fi inițiată de oricare din parteneri, o dată cu eliberarea conexiunii revenindu-se în starea *ÎNCHIS*.

Automatul cu stări finite este reprezentat în fig. 6-33. Cazul cel mai comun, al unui client conectându-se activ la un server pasiv, este reprezentat prin linii groase - continue pentru client și întrerupte pentru server. Liniile subțiri reprezintă secvențe de evenimente mai puțin obișnuite, dar posibile. Fiecare linie din fig. 6-33 este etichetată cu o pereche *eveniment/acțiune*. Evenimentul poate fi unul inițiat de către utilizator printr-un apel sistem (CONNECT, LISTEN, SEND sau CLOSE), recepțio-

narea unui segment (*SYN, FIN, ACK* sau *RST*) sau, într-un singur caz, expirarea unui interval de timp egal cu dublul ciclului de viață a unui pachet. Acțiunea constă în expedierea unui segment de control (*SYN, FIN* sau *RST*) sau „nici o acțiune”, lucru reprezentat prin —. Comentariile sunt incluse între paranteze.

Diagrama poate fi înțeleasă cel mai bine urmărind de la bun început calea urmată de un client (linia groasă continuă) și apoi calea urmată de un server (linia groasă întreruptă). Atunci când un program aplicație de pe mașina client generează o cerere CONNECT, entitatea TCP locală creează o înregistrare de conexiune, o marchează ca fiind în starea *SYN SENT* și trimite un segment *SYN*. De observat că mai multe conexiuni pot fi deschise (sau în curs de a fi deschise) în același timp spre folosul mai multor aplicații, astfel încât o stare este asociată unei conexiuni și este înregistrată în înregistrarea asociată acesteia. La recepția unui *SYN + ACK*, TCP expediază ultima confirmare (*ACK*) din „înțelegerea în trei pași” și comută în starea *STABILIT*. Din acest moment, informația poate fi atât expediată cât și recepționată.

Atunci când se termină o aplicație, se apelează primitiva CLOSE care impune entității TCP locale expedierea unui segment *FIN* și așteptarea *ACK*-ului corespunzător (dreptunghiul figurat cu linie întreruptă și etichetat „închidere activă”). Atunci când *ACK*-ul este recepționat, se trece în starea *AȘTEPTARE FIN 2,* unul din sensuri fiind în acest moment închis. Atunci când celălalt sens este la rândul său închis de partenerul de conexiune, se recepționează un *FIN* care este totodată și confirmat. În acest moment, ambele sensuri sunt închise, dar TCP-ul așteaptă un interval de timp egal cu dublul duratei de viață a unui pachet, garantând astfel că toate pachetele acestei conexiuni au murit și că nici o confirmare nu a fost pierdută. Odată ce acest interval de timp expiră, TCP-ul șterge înregistrarea asociată conexiunii.

Să examinăm acum gestiunea conexiunii din punctul de vedere al server-ului. Acesta execută LISTEN și se „așează” fiind totodată atent pentru a vedea cine „se ridică în picioare”. La recepționarea unui *SYN*, acesta este confirmat și serverul comută în starea *SYN RCVD*. Atunci când *SYN*-ul server-ului este la rândul său confirmat, „înțelegerea în trei pași” este completă, serverul comutând în starea *STABILIT*. De acum, transferul informației poate începe.

Atunci când clientul a terminat, execută CLOSE, ceea ce conduce la atenționarea server-ului prin recepționarea unui *FIN* (dreptunghiul figurat cu linie întreruptă și etichetat „închidere pasivă”). Atunci când și acesta execută un CLOSE, se trimite un *FIN* către client. O dată cu primirea confirmării clientului, serverul desființează conexiunea și șterge înregistrarea asociată.

6.5.8 Politica TCP de transmisie a datelor

Cum s-a menționat anterior, administrarea ferestrei în TCP nu este direct legată de confirmări, așa cum se întâmplă la cele mai multe protocoale de nivel legătură de date. De exemplu, să presupunem că receptorul are un tampon de 4096 octeți, așa cum se vede în fig. 6-34. Dacă emițătorul transmite un segment de 2048 de octeți care este recepționat corect, receptorul va confirma segmentul. Deoarece acum tamponul acestuia din urmă mai are liberi doar 2048 octeți (până când aplicația șterge niște date din acest tampon), receptorul va anunța o fereastră de 2048 octeți începând de la următorul octet așteptat.

Acum, emițătorul transmite alți 2048 octeți, care sunt confirmați, dar fereastra oferită este 0. Emițătorul trebuie să se oprească până când procesul aplicație de pe mașina receptoare a șters niște date din tampon, moment în care TCP poate oferi o fereastră mai mare.

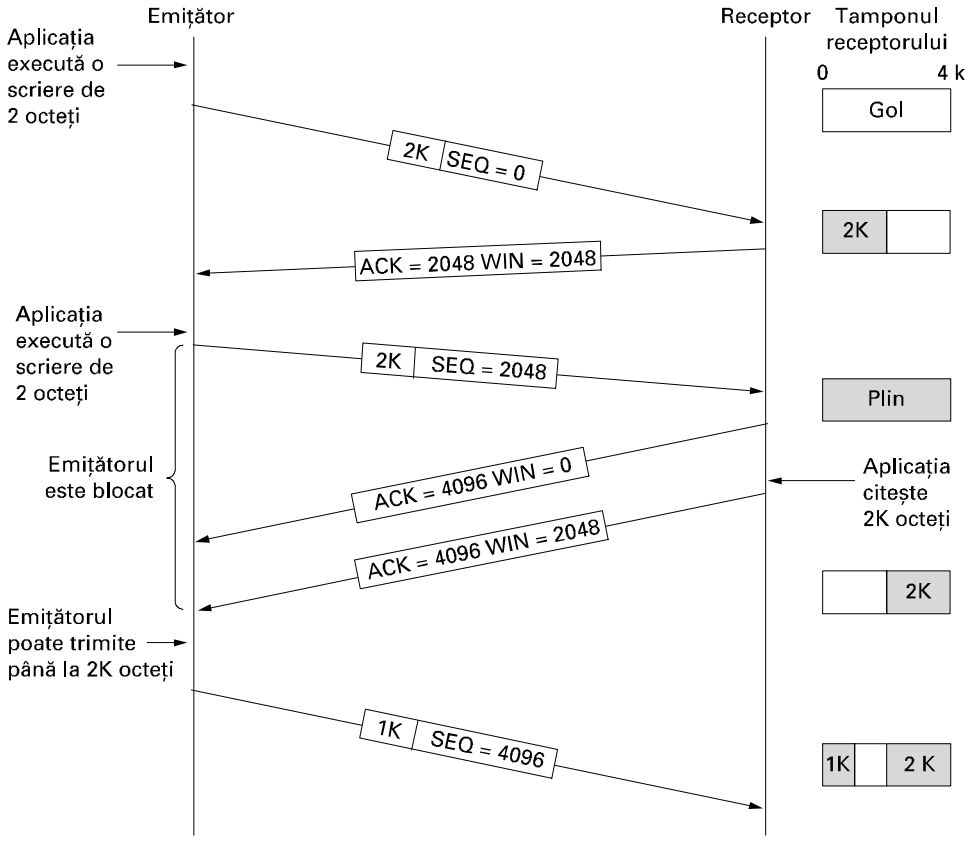


Fig. 6-34. Controlul ferestrei în TCP.

Atunci când fereastra este 0, în mod normal emițătorul nu poate să transmită segmente, cu două excepții. În primul rând, informația urgentă poate fi trimisă, de exemplu pentru a permite utilizatorului să oprească procesele rulând pe mașina de la distanță. În al doilea rând, emițătorul poate trimite un segment de un octet pentru a determina receptorul să renunțe următorul octet așteptat și dimensiunea ferestrei. Standardul TCP prevede în mod explicit această opțiune pentru a preveni interblocarea în cazul în care se întâmplă ca anunțarea unei ferestre să fie vreodată pierdută.

Emițătorii nu transmit în mod obligatoriu date de îndată ce acest lucru este cerut de către aplicație. Nici receptorii nu trimit în mod obligatoriu confirmările de îndată ce acest lucru este posibil. De exemplu, în fig. 6-34, atunci când sunt disponibili primii 2K octeți, TCP, știind că dispune de o fereastră de 4K octeți, va memora informația în tampon până când alți 2K octeți devin disponibili și astfel se va putea transmite un segment cu o încărcare utilă de 4K octeți. Această facilitate poate fi folosită pentru îmbunătățirea performanțelor.

Să considerăm o conexiune TELNET cu un editor interactiv care reacționează la fiecare apăsare a tastelor. În cel mai rău caz, atunci când un caracter sosește la entitatea TCP emițătoare, TCP creează un segment TCP de 21 octeți, pe care îl furnizează IP-ului pentru a fi transmis ca o datagramă IP de 41 octeți. De partea receptorului, TCP transmite imediat o confirmare de 40 octeți (20 octeți antet TCP și 20 octeți antet IP). Mai târziu, când editorul a citit caracterul, TCP transmite o

actualizare a ferestrei, deplasând fereastra cu un octet la dreapta. Acest pachet este de asemenea de 40 octeți. În final, când editorul a prelucrat caracterul, transmite ecoul sub forma unui pachet de 41 octeți. Cu totul, sunt folosiți 162 octeți din lărgimea de bandă și sunt trimise patru segmente pentru orice caracter tipărit. Atunci când lărgimea de bandă este redusă, această metodă de lucru nu este recomandată.

O abordare folosită de multe implementări TCP pentru optimizarea acestei situații constă în întârzierea confirmărilor și actualizărilor de fereastră timp de 500 ms, în speranța apariției unor informații la care să se atașeze pentru o călătorie pe gratis. Presupunând că editorul are un ecou de 50 ms, este necesar acum un singur pachet de 41 octeți pentru a fi trimis utilizatorului de la distanță, reducând numărul pachetelor și utilizarea lărgimii de bandă la jumătate.

Deși această regulă reduce încărcarea rețelei de către receptor, emițătorul operează încă ineficient trimițând pachete de 41 octeți care conțin un singur octet de date. O modalitate de a reduce această deficiență este cunoscută ca **algoritmul lui Nagle** (Nagle, 1984). Sugestia lui Nagle este simplă: atunci când emițătorul dispune de date, în secvență, de câte un octet, el va trimite doar primul octet, restul octeților fiind memorați până la confirmarea primului octet. Apoi vor fi trimise toate caracterele memorate într-un segment TCP și va continua memorarea până la confirmarea tuturor octeților. Dacă utilizatorul tastează repede și rețeaua este lentă, un număr substanțial de caractere poate fi plasat în fiecare segment, reducând cu mult lărgimea de bandă folosită. În plus, algoritmul permite transmisia unui nou pachet, dacă s-a dispus de suficientă informație pentru a umple jumătate dintr-o fereastră sau pentru a completa un segment.

Implementările TCP folosesc pe scară largă algoritmul lui Nagle, dar există situații când este mai bine ca el să fie dezactivat. În particular, când o aplicație X-Windows rulează prin Internet, deplasările mausului trebuie transmise mașinii de la distanță. (Sistemul X Window este sistemul de ferestre utilizat pe majoritatea sistemelor UNIX.) Gruparea lor pentru a fi transmise în rafală provoacă o mișcare imprevizibilă a cursorului, lucru care nemulțumește profund utilizatorii.

O altă problemă care poate degrada performanța TCP este **sindromul ferestrei stupide**. (Clark, 1982). Această problemă apare atunci când informația este furnizată entității TCP emițătoare în blocuri mari, dar la partea receptoare o aplicație interactivă citește datele octet cu octet. Pentru a înțelege problema, să analizăm fig. 6-35. Inițial, tamponul TCP al receptorului este plin și emițătorul știe acest fapt (adică are o fereastră de dimensiune 0). Apoi, aplicația interactivă citește un caracter de pe canalul TCP. Această acțiune face fericită entitatea TCP receptoare, deci ea va trimite o actualizare de fereastră către emițător dându-i astfel dreptul de a mai trimite un octet. Îndatorat, emițătorul trimite un octet. Cu acesta, tamponul este plin și receptorul confirmă segmentul de 1 octet, dar repoziționează dimensiunea ferestrei la 0. Acest comportament poate continua la nesfârșit.

Soluția lui Clark este de a nu permite receptorului să trimită o actualizare de fereastră la fiecare octet. În schimb, el este forțat să aștepte până când spațiul disponibil are o dimensiune decentă, urmând să-l ofere pe acesta din urmă. Mai precis, receptorul nu ar trebui să trimită o actualizare de fereastră până când nu va putea gestiona minimul dintre dimensiunea maximă oferită atunci când conexiunea a fost stabilită și jumătate din dimensiunea tamponului său, dacă este liberă.

Mai mult decât atât, emițătorul poate îmbunătăți situația netrimițând segmente de dimensiune mică. În schimb, el ar trebui să încerce să aștepte până când acumulează suficient spațiu în fereastră pentru a trimite un segment întreg sau măcar unul conținând cel puțin jumătate din dimensiunea tamponului receptorului (pe care trebuie să o estimeze din secvența actualizărilor de fereastră recepționate până acum).

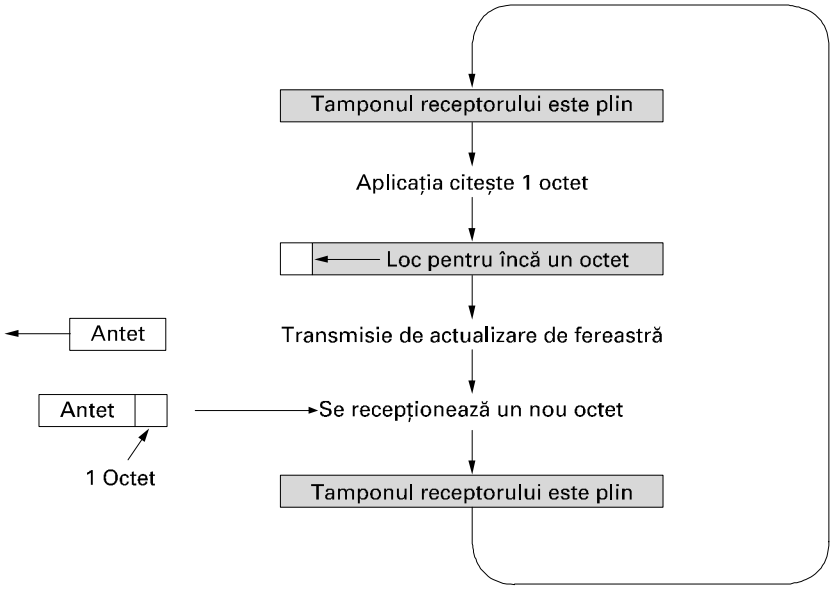


Fig. 6-35. Sindromul ferestrei stupide.

Algoritmul lui Nagle și soluția lui Clark pentru sindromul ferestrei stupide sunt complementare. Nagle a încercat să rezolve problema furnizării datelor către TCP octet cu octet, cauzată de aplicația emițătoare. Clark a încercat să rezolve problema extragerii datelor de la TCP octet cu octet, cauzată de către aplicația receptoare. Ambele soluții sunt valide și pot funcționa împreună. Scopul este ca emițătorul să nu trimită segmente mici, iar receptorul să nu ceară astfel de segmente.

Receptorul TCP poate face, pentru îmbunătățirea performanțelor, mai mult decât simpla actualizare a ferestrei în unități mari. Ca și emițătorul TCP, el are posibilitatea să memoreze date, astfel încât să poată bloca o cerere de READ a aplicației până când îi poate furniza o cantitate semnificativă de informație. Astfel se reduce numărul de apeluri TCP, deci și supraîncărcarea. Bineînțeles, în acest mod va crește și timpul de răspuns, dar pentru aplicații care nu sunt interactive, așa cum este transferul de fișiere, eficiența poate fi mai importantă decât timpul de răspuns la cereri individuale.

O altă problemă de discutat despre receptor se referă la ce trebuie să facă acesta cu segmentele care nu sosesc în ordine. Ele pot fi reținute sau eliminate, după cum dorește receptorul. Bineînțeles, confirmările pot fi trimise numai atunci când toată informația până la octetul confirmat a fost recepționată. Dacă receptorul primește segmentele 0, 1, 2, 4, 5, 6 și 7, el poate confirma totul până la ultimul octet din segmentul 2 inclusiv. Atunci când emițătorul constată o depășire de timp, el va retransmite segmentul 3. Dacă receptorul a memorat în tampon segmentele 4 până la 7, odată cu recepția segmentului 3 el poate confirma toți octeții până la sfârșitul segmentului 7.

6.5.9 Controlul congestiei în TCP

Atunci când încărcarea la care este supusă o rețea este mai mare decât poate aceasta să suporte, apare congestia. Internet-ul nu face excepție. În această secțiune, vom discuta algoritmi care se ocupă cu astfel de congestii și care au fost dezvoltați pe parcursul ultimului sfert de secol. Deși nivelul rețea

încearcă de asemenea să controleze congestia, cea mai mare parte a muncii este făcută de TCP, și aceasta deoarece adevărata soluție a congestiei constă în micșorarea ratei de transfer a informației.

Teoretic, congestia poate fi controlată pe baza unui principiu împrumutat din fizică: legea conservării pachetelor. Ideea de bază este de a nu injecta un nou pachet în rețea până când un pachet mai vechi nu o părăsește (de exemplu este furnizat receptorului). TCP încearcă să atingă acest scop prin manipularea dinamică a dimensiunii ferestrei.

Primul pas în controlul congestiei este detecția ei. Mai demult, detecția congestiei era dificilă. O depășire de timp datorată pierderii unui pachet putea fi cauzată fie de (1) zgomotul de pe linia de transmisie, fie de (2) eliminarea pachetului de către un ruter congestionat. Diferențierea celor două cazuri era dificilă.

În zilele noastre, pierderea pachetului din pricina erorilor de transmisie este destul de rară, deoarece cele mai multe din trunchiurile principale de comunicație sunt din fibră (deși rețelele fără fir sunt un subiect separat). În consecință, cele mai multe depășiri ale timpilor de transmisie pe Internet se datorează congestiilor. Toți algoritmii TCP din Internet presupun că depășirile de timp sunt cauzate de congestii și monitorizează aceste depășiri pentru a detecta problemele.

Înainte de a discuta despre modalitatea în care TCP reacționează la congestii, să descriem în primul rând modul în care se încearcă prevenirea apariției lor. Atunci când se stabilește o conexiune, trebuie să se aleagă o fereastră de o dimensiune potrivită. Receptorul poate specifica o fereastră bazându-se pe dimensiunea tamponului propriu. Dacă emițătorul acceptă această dimensiune a ferestrei, nu mai pot apărea probleme datorită depășirii tamponului la recepție, dar pot apărea în schimb datorită congestiei interne în rețea.

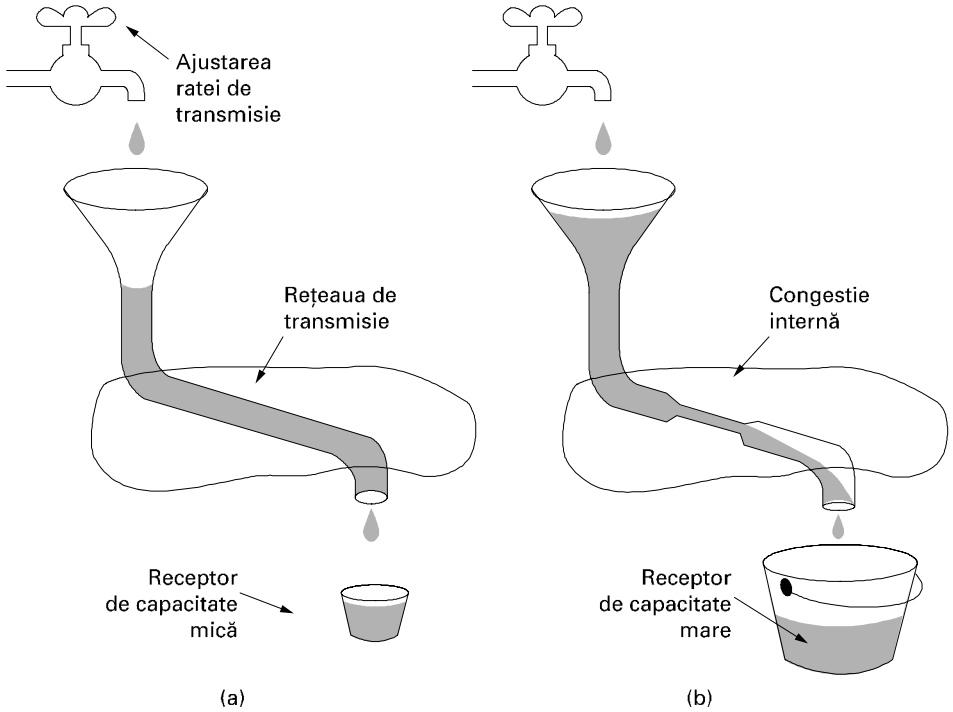


Fig. 6-36. (a) O rețea rapidă alimentând un rezervor de capacitate mică. (b) O rețea lentă alimentând un receptor de mare capacitate.

În fig. 6-36, putem vedea interpretarea hidraulică a acestei probleme. În fig. 6-36(a), observăm o conductă groasă care duce la un receptor de dimensiune mică. Atâta timp cât emițătorul nu trimite mai multă apă decât poate conține găleata, apa nu se va pierde. În fig. 6-36(b), factorul de limitare nu mai este capacitatea găleții, ci capacitatea de transport a rețelei. Dacă vine prea repede prea multă apă, ea se va revărsa și o anumită cantitate se va pierde (în acest caz prin umplerea pâlniei).

Soluția din Internet este de a realiza posibilitatea existenței a două probleme - capacitatea rețelei și capacitatea receptorului - și de a le trata pe fiecare separat. Pentru a face acest lucru, fiecare emițător menține două ferestre: fereastra acceptată de către receptor și o a doua fereastră, **fereastra de congestie**. Fiecare reflectă numărul de octeți care pot fi trimiși de către emițător. Numărul octeților care pot fi trimiși este dat de minimul dintre cele două ferestre. Astfel, fereastra efectivă este minimul dintre ceea ce emițătorul crede că este „în regulă" și ceea ce receptorul crede că este „în regulă". Dacă receptorul spune: „Trimite 8K octeți", dar emițătorul știe că o rafală de mai mult de 4K octeți poate aglomera excesiv rețeaua, el va trimite 4K octeți. Din alt punct de vedere, dacă receptorul spune: „Trimite 8K octeți" și emițătorul știe că o rafală de 32K octeți poate străbate fără efort rețeaua, el va trimite toți cei 8K octeți ceruți.

La stabilirea conexiunii, emițătorul inițializează fereastra de congestie la dimensiunea celui mai mare segment utilizat de acea conexiune. El trimite apoi un segment de dimensiune maximă. Dacă acest segment este confirmat înaintea expirării timpului, mai adaugă un segment la fereastra de congestie, făcând-o astfel de dimensiunea a două segmente de dimensiune maximă, și trimite două segmente. O dată cu confirmarea fiecăruia din aceste segmente, fereastra de congestie este redimensionată cu încă un segment de dimensiune maximă. Atunci când fereastra de congestie este de n segmente, dacă toate cele n segmente sunt confirmate în timp util, ea este crescută cu numărul de octeți corespunzător celor n segmente. De fapt, fiecare rafală confirmată cu succes dublează fereastra de congestie.

Fereastra de congestie crește în continuare exponențial până când sau se produce o depășire de timp, sau se atinge dimensiunea ferestrei receptorului. Ideea este ca dacă rafale de dimensiune, să spunem, 1024, 2048 și 4096 de octeți funcționează fără probleme, dar o rafală de 8192 octeți duce la o depășire de timp, fereastra de congestie va fi stabilită la 4096 de octeți pentru a evita congestia. Atâta timp cât fereastra de congestie rămâne la 4096, nu va fi transmisă nici o rafală mai mare de această valoare, indiferent cât de mult spațiu de fereastră este oferit de către receptor. Acest algoritm este numit **algoritmul startului lent**, fără a fi însă câtuși de puțin lent (Jacobson, 1988). Este exponențial. Toate implementările TCP trebuie să îl suporte.

Să privim acum algoritmul de control al congestiei în cazul Internetului. El utilizează în plus față de ferestrele de recepție și de congestie un al treilea parametru, numit **prag**, inițial de 64K. Atunci când apare o depășire de timp, pragul este poziționat la jumătate din fereastra curentă de congestie și fereastra de congestie este repoziționată la dimensiunea unui segment maxim. Startul lent este utilizat apoi pentru a determina cât poate rețeaua să ducă, atâta doar că acea creștere exponențială se oprește odată cu atingerea pragului. De aici înainte transmisiile reușite măresc în mod liniar dimensiunea ferestrei de congestie (cu câte un segment maxim pentru fiecare rafală), în locul unei creșteri pentru fiecare segment. De fapt, algoritmul presupune că este acceptabilă înjumătățirea ferestrei de congestie, din acel punct continuându-și gradual calea spre dimensiuni mai mari.

Funcționarea algoritmului de congestie se poate vedea în fig. 6-37. Dimensiunea unui segment maxim este, în acest caz, de 1024 de octeți. Inițial fereastra de congestie are 64K octeți, dar apare o depășire de timp și deci pragul este stabilit la 32K octeți iar fereastra de congestie la 1K octeți acesta fiind punctul 0 al transmisiei din figură. Fereastra de congestie crește apoi exponențial până atinge pragul (32K octeți). Începând de aici, creșterea este liniară.

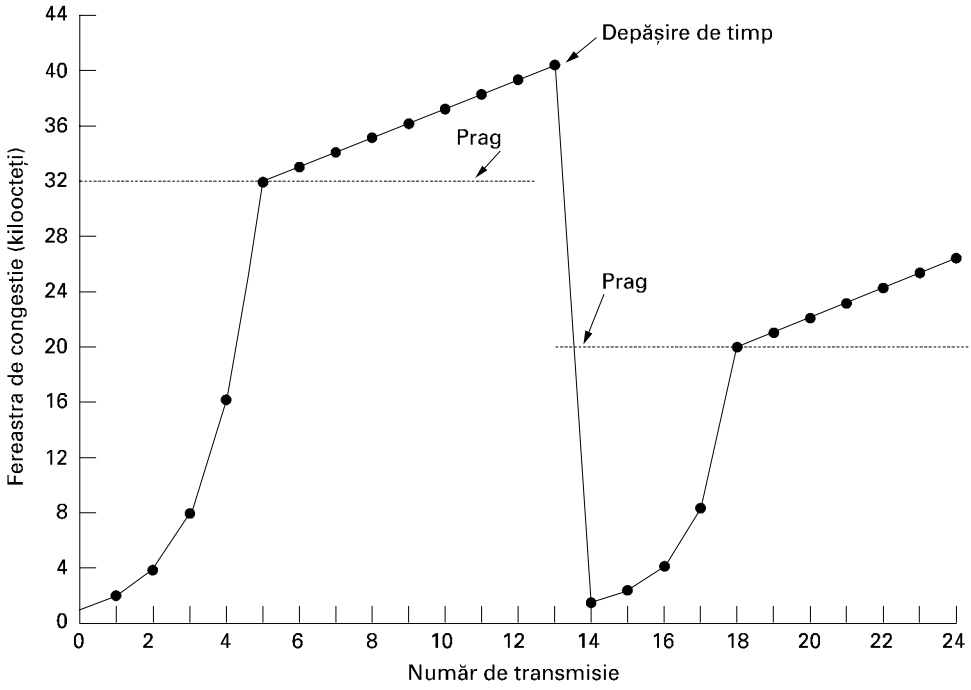


Fig. 6-37. Un exemplu al algoritmului de congestie din Internet.

Transmisia 13 nu are noroc (este de la sine înțeles) și apare o depășire de timp. Pragul este stabilit la jumătate din fereastra curentă (acum 40K octeți, deci jumătate este 20K octeți) și startul lent este inițiat din nou. Atunci când confirmările pentru transmisia 14 încep să sosească, primele patru dublează fiecare fereastra de congestie, dar după aceea creșterea redevine liniară.

Dacă nu mai apar depășiri de timp, fereastra de congestie va continua să crească până la dimensiunea ferestrei receptorului. În acest punct, creșterea ei va fi oprită și va rămâne constantă atâta timp cât nu mai apar depășiri și fereastra receptorului nu își modifică dimensiunea. Ca un alt aspect, dacă un pachet ICMP SOURCE QUENCH sosește și este furnizat TCP-ului, acest eveniment este tratat la fel ca o depășire de timp. O alternativă (și o abordare mai recentă) este descrisă în RFC 3168.

6.5.10 Administrarea contorului de timp în TCP

TCP utilizează (cel puțin conceptual) mai multe contoare pentru a face ceea ce are de făcut. Cel mai important dintre acestea este **contorul de retransmisie**. Atunci când este trimis un segment, se pornește un contor de retransmisie. Dacă segmentul este confirmat înainte de expirarea timpului, contorul este oprit. Pe de altă parte, dacă timpul expiră înaintea primirii confirmării, segmentul este retransmis (și contorul este pornit din nou). Întrebarea care se pune este următoarea: Cât de mare trebuie să fie intervalul de timp până la expirare?

Această problemă este mult mai dificilă la nivelul transport din Internet decât la nivelul protocoalelor generice de legătură de date prezentate în Cap. 3. În cazul din urmă, întârzierea așteptată este ușor predictibilă (de exemplu, are o variantă scăzută), deci contorul poate fi setat să expire chiar

imediat după ce era așteptată confirmarea, așa cum se arată în fig. 6-38(a). Cum confirmările sunt rareori întârziate în nivelul legătură de date, absența unei confirmări în momentul în care aceasta era așteptată înseamnă de obicei că s-a pierdut fie cadrul, fie confirmarea.

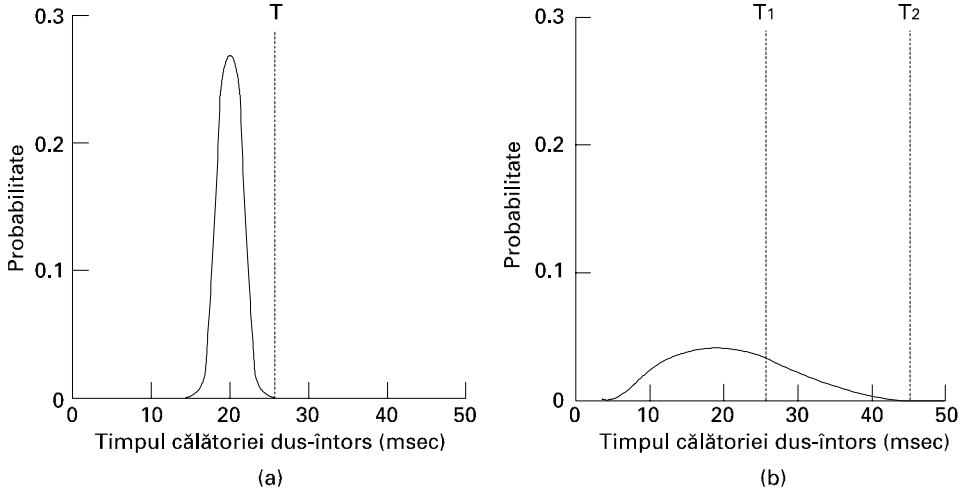


Fig. 6-38. (a) Densitatea de probabilitate a sosirilor în timp a confirmărilor în nivelul legătură de date. (b) Densitatea de probabilitate a sosiri confirmărilor pentru TCP.

TCP trebuie să facă față unui mediu radical diferit. Funcția de densitate a probabilității pentru timpul necesar întoarcerii unei confirmări TCP arată mai degrabă ca în fig. 6-38(b) decât ca în fig. 6-38(a). Este dificilă determinarea timpului în care se realizează circuitul dus-întors până la destinație. Chiar și când acesta este cunoscut, stabilirea intervalului de depășire este de asemenea dificilă. Dacă intervalul este prea scurt, să spunem T_1 în fig. 6-38(b), vor apărea retransmisii inutile, aglomerând Internet-ul cu pachete fără rost. Dacă este prea lung, (T_2), performanțele vor avea de suferit datorită întârzierii retransmisiei de fiecare dată când se pierde un pachet. Mai mult decât atât, media și varianța distribuției sosirii confirmărilor se pot schimba cu rapiditate pe parcursul a câtorva secunde atunci când apare sau se rezolvă o congestie.

Soluția este să se utilizeze un algoritm profund dinamic, care ajustează în mod constant intervalul de depășire bazându-se pe măsurători continue ale performanței rețelei. Algoritmul utilizat în general de către TCP este datorat lui Jacobson (1988) și este descris mai jos. Pentru fiecare conexiune, TCP păstrează o variabilă, *RTT*, care este cea mai bună estimare a timpului în care se parcurge circuitul dus-întors către destinația în discuție. Atunci când este trimis un segment, se pornește un contor de timp, atât pentru a vedea cât de mult durează până la primirea confirmării cât și pentru a inița o retransmisie în cazul scurgerii unui interval prea lung. Dacă se primește confirmarea înaintea expirării contorului, TCP măsoară cât de mult i-a trebuit confirmării să sosească, fie acest timp *M*. În continuare el actualizează *RTT*, după formula:

$$RTT = \alpha RTT + (1 - \alpha)M$$

unde α este un factor de netezire care determină ponderea dată vechii valori. Uzual, $\alpha = 7/8$.

Chiar presupunând o valoare bună a lui *RTT*, alegerea unui interval potrivit de retransmisie nu este o sarcină ușoară. În mod normal, TCP utilizează βRTT, dar problema constă în alegerea lui β.

În implementările inițiale, β era întotdeauna 2, dar experiența a arătat că o valoare constantă este inflexibilă deoarece nu corespunde în cazul creșterii variației.

În 1988, Jacobson a propus ca β să fie aproximativ proporțional cu deviația standard a funcției de densitate a probabilității timpului de primire a confirmărilor, astfel încât o variație mare implică un β mare și viceversa. În particular, el a sugerat utilizarea *deviației medii* ca o estimare puțin costisitoare a *deviației standard*. Algoritmul său presupune urmărirea unei alte variabile de netezire, D, deviația. La fiecare sosire a unei confirmări, este calculată diferența dintre valorile așteptate și observate $|RTT - M|$. O valoare netezită a acesteia este păstrată în D, prin formula

$$D = \alpha D + (1 - \alpha)|RTT - M|$$

unde α poate sau nu să aibă aceeași valoare ca cea utilizată pentru netezirea lui *RTT*. Deși *D* nu este chiar deviația standard, ea este suficient de bună și Jacobson a arătat cum poate fi calculată utilizând doar adunări de întregi, scăderi și deplasări, ceea ce este un mare punct câștigat. Cele mai multe implementări TCP utilizează acum acest algoritm și stabilesc valoarea intervalului de depășire la:

$$\text{Depășire} = RTT + 4 * D$$

Alegerea factorului 4 este într-un fel arbitrară, dar are două avantaje. În primul rând, multiplicarea prin 4 poate fi făcută printr-o singură deplasare. În al doilea rând, minimizează depășirile de timp și retransmisiile inutile, datorită faptului că mai puțin de un procent din totalul pachetelor sosesc cu întârzieri mai mari de patru ori deviația standard. (De fapt, Jacobson a propus inițial să se folosească 2, dar experiența ulterioară a demonstrat că 4 conduce la performanțe mai bune).

O problemă legată de estimarea dinamică a *RTT* se referă la ce trebuie făcut în situația în care un segment cauzează o depășire și trebuie retransmis. Atunci când confirmarea este primită, nu este clar dacă aceasta se referă la prima transmisie sau la o transmisie următoare. O decizie eronată poate contamina serios estimarea lui *RTT*. Phil Karn a descoperit această problemă cu multă greutate. El este un radio amator entuziast, interesat în transmiterea pachetelor TCP/IP prin operare radio, un mediu recunoscut pentru lipsa de fiabilitate (pe o zi senină, la destinație ajung jumătate din pachete). El a făcut o propunere simplă: să nu se actualizeze *RTT* pentru nici un segment care a fost retransmis. În loc de aceasta, timpul de expirare este dublat la fiecare eșec, până când segmentele ajung prima oară la destinație. Această corecție este numită **algoritmul lui Karn**. Cele mai multe din implementările TCP utilizează acest algoritm.

Contorul de retransmisie nu este singurul utilizat de către TCP. Un al doilea contor este **contorul de persistență**. El este proiectat să prevină următoarea situație de interblocare. Receptorul trimite o confirmare cu o dimensiune a ferestrei 0, spunându-i emițătorului să aștepte. Mai târziu, receptorul actualizează fereastra, dar pachetul cu această actualizare este pierdut. Acum, atât emițătorul cât și receptorul așteaptă o reacție din partea celuilalt. Atunci când contorul de persistență expiră, emițătorul transmite o sondă către receptor. Răspunsul la această investigare furnizează dimensiunea ferestrei. Dacă această dimensiune continuă să fie zero, contorul de persistență este repoziționat și ciclul se repetă. Dacă este nenul, informația poate fi acum transmisă.

Un al treilea contor utilizat de unele implementări este **contorul de menținere în viață**. Când o conexiune a fost inactivă o lungă perioadă de timp, contorul de menținere în viață poate expira pentru a forța una din părți să verifice dacă cealaltă parte există încă. Dacă aceasta nu răspunde, conexiunea este închisă. Această facilitate este controversată, deoarece supraîncarcă rețeaua și poate termina o conexiune altfel sănătoasă datorită unei partiționări tranzitorii a rețelei.

Ultimul contor folosit la fiecare conexiune TCP este acela utilizat în stările de *AȘTEPTARE CONTORIZATĂ* pe parcursul închiderii conexiunilor. El funcționează pe durata a două vieți maxime ale unui pachet, pentru a se asigura că, atunci când o conexiune este închisă, toate pachetele create de aceasta au murit.

6.5.11 TCP și UDP în conexiune fără fir

Teoretic, protocoalele de transport ar trebui să fie independente de tehnologia nivelului rețea. În particular, TCP-ului ar trebui să nu-i pese dacă IP rulează peste o rețea cablu sau radio. În practică, acest lucru contează, deoarece cele mai multe implementări TCP au fost atent optimizate pe baza unor presupuneri care sunt adevărate pentru rețele cu cabluri, dar care nu mai sunt valabile în cazul rețelelor fără fir. Ignorarea proprietăților de transmisie fără fir poate conduce la o implementare TCP corectă din punct de vedere logic, dar cu performanțe incredibil de proaste.

Principala problemă este algoritmul de control al congestiei. Aproape toate implementările TCP din zilele noastre pleacă de la premisa că depășirile de timp sunt cauzate de congestie și nu de pierderea pachetelor. În consecință, atunci când expiră un contor, TCP încetinește ritmul și trimite pachete cu mai puțină vigoare (ex. algoritmul startului lent al lui Jacobson). Ideea din spatele acestei abordări constă în reducerea încărcării rețelei, astfel eliminându-se neplăcerile cauzate de congestie.

Din nefericire, legăturile bazate pe transmisia fără fir nu sunt deloc fiabile. Ele pierd tot timpul pachete. Pentru a controla această pierdere a pachetelor, abordarea corectă este să se retrimită cât mai repede posibil. Încetinirea ritmului nu face decât să înrăutățească lucrurile. Dacă presupunem că, atunci când emițătorul transmite 100 de pachete pe secundă, 20% din totalul pachetelor se pierde, productivitatea este de 80 pachete/sec. Dacă emițătorul încetinește ritmul la 50 pachete/sec, productivitatea scade la 40 pachete/sec.

Atunci când se pierde un pachet pe o rețea cu cabluri, emițătorul ar trebui să încetinească ritmul. Atunci când se pierde un pachet pe o rețea fără fir, emițătorul ar trebui să îl mărească. Dacă emițătorul nu știe despre ce tip de rețea este vorba, luarea unei decizii este dificilă.

În mod frecvent, calea de la emițător la receptor este eterogenă. Primii 1000 km pot să fie într-o rețea cu cabluri, dar ultimul kilometru poate să fie fără fir. Acum, luarea unei decizii în situația unei depășiri de timp este și mai dificilă, dat fiind că intervine și locul în care apare problema. O soluție propusă de Bakne și Badrinath (1995), **TCP indirect**, constă în spargerea conexiunii TCP în două conexiuni separate, ca în fig. 6-39. Prima conexiune pleacă de la emițător la stația de bază. Cea de-a doua leagă stația de bază de receptor. Această stație de bază nu face decât să copieze pachetele din cele două conexiuni în ambele direcții.

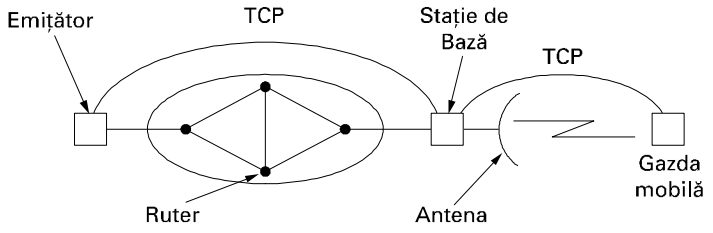


Fig. 6-39. Spargerea conexiunii TCP în două conexiuni.

Avantajul acestei scheme este acela că ambele conexiuni sunt acum omogene. Depășirile de timp din prima conexiune pot încetini emițătorul, în timp ce depășirile de timp din cea de-a doua îl pot accelera. Alți parametri pot fi de asemenea reglați separat în fiecare din cele două conexiuni. Dezavantajul este acela că este negată însăși semantica TCP. Atâta timp cât fiecare parte a conexiunii este o conexiune TCP în sine, stația de bază confirmă fiecare segment TCP în mod obișnuit. Doar că acum, recepția unei confirmări de către emițător nu mai înseamnă că receptorul a primit segmentul, ci doar că el a fost primit de către stația de bază.

O soluție diferită, datorată lui Balakrishnan et. al (1995), nu încalcă semantica TCP. Ea se bazează pe mici modificări făcute în codul nivelului rețea din stația de bază. Una din modificări constă în adăugarea unui agent de supraveghere care observă și interceptează pachetele TCP care pleacă spre gazda mobilă precum și confirmările care se întorc de la acesta. Atunci când observă un segment TCP care pleacă spre gazda mobilă, dar nu observă confirmarea recepționării acestuia într-un interval de timp dat (relativ scurt), agentul ascuns pur și simplu retransmite acel segment, fără a mai spune sursei acest lucru. De asemenea, el retransmite și atunci când observă confirmări duplicate din partea gazdei mobile, lucru care indică invariabil faptul că aceasta a pierdut ceva. Confirmările duplicate sunt eliminate pe loc, pentru a evita ca sursa să le interpreteze ca un semn de congestie.

Cu toate acestea, un dezavantaj al acestei transparențe este acela că, dacă legătura fără fir pierde multe pachete, sursa poate depăși limita de timp în așteptarea unei confirmări și poate invoca în consecință algoritmul de control al congestiei. În cazul TCP-ului indirect, algoritmul de control al congestiei nu va fi niciodată inițiat dacă nu apare într-adevăr o situație de congestie în partea „cablată" a rețelei.

Algoritmul Balakrishnan oferă de asemenea o soluție problemei pierderii segmentelor generate de către gazda mobilă. Atunci când stația de bază constată o pauză în interiorul domeniului numerelor de secvență, aceasta generează o cerere pentru o repetare selectivă a octetului lipsă, utilizând o opțiune TCP.

Utilizând aceste corecturi, legătura fără fir devine mai fiabilă în ambele direcții fără ca sursa să știe acest lucru și fără modificarea semanticii TCP.

Deși UDP-ul nu suferă de aceleași probleme ca și TCP-ul, comunicația fără fir induce și pentru el anumite dificultăți. Principala problemă este aceea că programele utilizează UDP se așteaptă ca acesta să fie foarte fiabil. Ele știu că nu este furnizată nici o garanție, dar cu toate acestea se așteaptă ca el să fie aproape perfect. Într-un mediu fără fir, el va fi însă departe de perfecțiune. Pentru programele care sunt capabile să se refacă după pierderea mesajelor UDP, dar numai cu un cost considerabil, trecerea bruscă de la un mediu în care mesajele puteau fi pierdute mai mult teoretic decât practic la un mediu în care ele sunt pierdute sistematic poate conduce la un dezastru în ceea ce privește performanțele.

Comunicația fără fir afectează și alte domenii decât cel al performanțelor. De exemplu, cum poate o gazdă mobilă să găsească o imprimantă locală la care să se conecteze, în loc să utilizeze propria imprimantă? Oarecum legată de aceasta este și problema obținerii paginii WWW pentru celula locală, chiar dacă numele ei nu este cunoscut. De asemenea, proiectanții paginilor WWW au tendința să presupună disponibilă o mare lărgime de bandă. Punerea unei embleme mari pe fiecare pagină poate să devină contraproductivă dacă transmisia paginii printr-o legătură fără fir lentă va dura 10 secunde, și acest lucru ajunge până la urmă să irite utilizatorii.

Cum rețelele cu comunicații fără fir devin tot mai comune, problema rulării TCP-ului pe ele a devenit tot mai acută. Documentații suplimentare în acest domeniu se găsesc în (Barakat ș.a., 2000; Ghani și Dixit, 1999; Huston, 2001; și Xylomenos ș.a., 2001).

6.5.12 TCP Tranzacțional

Mai devreme în acest capitol am analizat apelul de proceduri la distanță ca modalitate de a implementa sistemele client-server. Dacă atât cererea, cât și răspunsul sunt suficient de mici încât să se potrivească în pachete simple și operația este idempotentă, UDP-ul poate fi ușor utilizat. Totuși, dacă aceste condiții nu sunt îndeplinite, utilizarea UDP-ului este mai puțin atractivă. De exemplu, dacă răspunsul este unul lung, atunci datagramele trebuie sa fie secvențiate și trebuie inițiat un mecanism pentru a retransmite datagramele pierdute. De fapt, aplicației îi este cerut să reinventeze TCP-ul.

În mod cert, acest lucru nu este atractiv, dar nici utilizarea TCP-ului în sine nu este atractivă. Problema este eficiența. Secvența normală a pachetelor pentru a face un RPC peste TCP este prezentată în fig. 6-40(a). În cel mai bun caz sunt necesare nouă pachete.

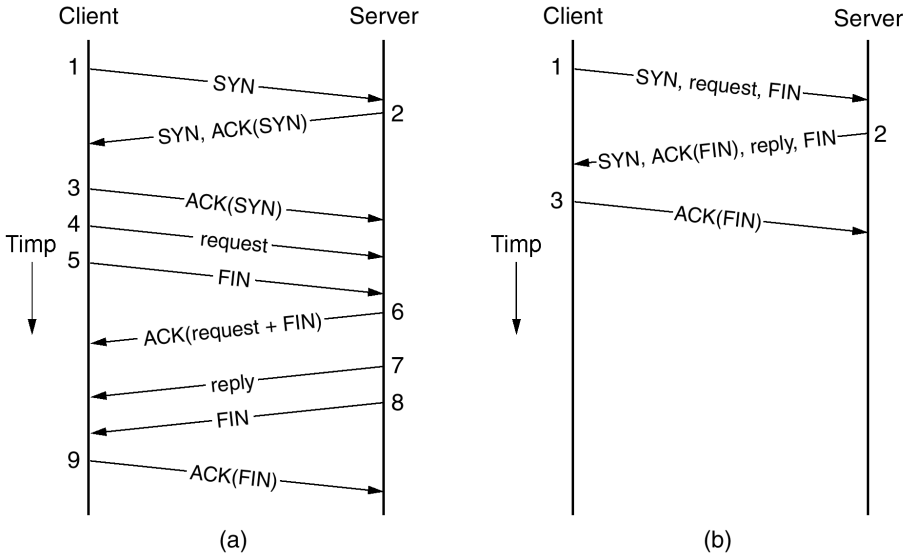


Fig. 6-40. (a) RPC folosind TCP clasic ; (b) RPC folosind T/TCP

Cele nouă pachete sunt după cum urmează:

1. Clientul trimite un pachet SYN pentru a stabili o conexiune.
2. Serverul trimite un pachet ACK pentru a recunoaște pachetul SYN.
3. Clientul finalizează înțelegerea în trei pași.
4. Clientul trimite cererea reală.
5. Clientul trimite un pachet FIN pentru a indica dacă s-a terminat trimiterea.
6. Serverul confirmă cererea și FIN-ul.
7. Serverul trimite răspunsul înapoi clientului.
8. Serverul trimite un pachet FIN pentru a indica că și acest lucru s-a încheiat.
9. Clientul confirmă FIN-ul server-ului.

A se reține că acesta este cazul ideal. În cazul cel mai rău, cererea clientului și FIN-ul sunt confirmate separat, precum sunt răspunsul server-ului și FIN-ul.

Întrebarea care apare imediat este dacă există vreo posibilitate de a combina eficiența RPC-ului folosind UDP (doar 2 mesaje) cu fiabilitatea TCP-ului. Răspunsul este: aproape că da. Se poate

realiza cu o variantă experimentală de TCP numită **T/TCP** (**Transactional TCP**, rom: TCP tranzacțional), care este descrisă în RFC 1379 și 1644.

Ideea principală este aceea de a modifica secvența standard de inițializare a conexiunii astfel încât să permită, la nivel înalt, transferul de date în timpul inițializării. Protocolul T/TCP este prezentat în fig. 6-40(b). Primul pachet al clientului conține bitul SYN, cererea în sine și FIN-ul. De fapt acesta spune: vreau să stabilesc o conexiune, aici sunt datele și am terminat

Când serverul primește cererea, caută sau calculează răspunsul și alege modul în care să răspundă. Dacă răspunsul încape într-un pachet, dă răspunsul din fig. 6-40(b), care spune: confirm FIN-ul tău, iată răspunsul, iar eu am terminat. Clientul confirmă apoi FIN-ul server-ului și protocolul ia sfârșit în (după) trei mesaje.

În orice caz, dacă rezultatul este mai mare de 1 pachet, serverul are de asemenea și opțiunea de a nu seta bitul FIN, caz în care poate trimite pachete multiple înainte de a-i închide direcția.

Merită probabil menționat faptul că T/TCP nu este singura îmbunătățire propusă pentru TCP. O altă propunere este **SCTP** (**Stream Control Transmission Protocol**, rom: Protocolul de control al transmisiei fluxului). Caracteristicile sale includ păstrarea legăturilor dintre mesaje, modalități multiple de livrare (de ex: livrarea neordonată), găzduirea multiplă (destinații de rezervă), și confirmări selective (Stewart and Metz, 2001). În orice caz, oricând cineva propune să schimbe ceva care a funcționat atât de bine de atâta timp, se duce o luptă aprigă între tabăra "utilizatorii doresc mai multe facilități" și cea "dacă nu e stricat, nu-l repara!".

6.6 ELEMENTE DE PERFORMANȚĂ

În rețelele de calculatoare sunt foarte importante elementele de performanță. Atunci când sunt interconectate sute sau mii de calculatoare, au loc adesea interacțiuni complexe cu consecințe nebănuite. Această complexitate conduce în mod frecvent la performanțe slabe fără ca cineva să știe de ce. În secțiunile următoare vom examina mai multe elemente legate de performanța rețelei pentru a identifica tipurile de probleme și ce poate fi făcut pentru rezolvarea lor.

Din nefericire, înțelegerea performanței rețelei este mai degrabă o artă decât o știință. Este prea puțină teorie care stă la bază și de fapt aceasta nu folosește în situații practice. Cel mai bun lucru pe care îl putem face este să indicăm reguli rezultate dintr-o experiență îndelungată și să prezentăm exemple luate din lumea reală. Am amânat în mod intenționat această discuție după studiul nivelului transport din rețelele TCP, pentru a fi capabili să folosim TCP ca exemplu în diverse locuri.

Nivelul transport nu este singurul loc unde apar elemente legate de performanță. Am văzut unele elemente la nivelul rețea, în capitolul precedent. Cu toate acestea, nivelul rețea tinde să fie preocupat în mare măsură de rutare și controlul congestiei. Problemele mai generale, orientate spre sistem ,tind să fie legate de nivelul transport, așa că acest capitol este locul potrivit pentru a le examina.

În următoarele cinci secțiuni vom examina cinci aspecte de performanță ale rețelei:

1. Probleme de performanță.
2. Măsurarea performanței rețelei.
3. Proiectarea de sistem pentru performanțe mai bune.
4. Prelucrarea rapidă TPDU.
5. Protocoale pentru rețele viitoare de mare performanță.

Ca o paranteză, avem nevoie de un nume pentru unitățile schimbate de către entitățile de transport. Termenul TCP de segment este cel mai confuz și în acest context nu este niciodată utilizat în afara lumii TCP. Termenii uzuali ATM (CS-PDU, SAR-PDU și CPCS-PDU) sunt specifici doar pentru ATM. Pachetele se referă în mod clar la nivelul rețea și mesajele aparțin nivelului aplicație. Din lipsa unui termen standard, vom reveni la numirea unităților schimbate de entitățile de transport TPDU-uri. Când ne referim atât la TPDU cât și la pachet, vom utiliza pachet ca un termen comun, ca de exemplu în „Procesorul trebuie să fie suficient de rapid pentru a prelucra pachetele în timp real." Prin aceasta subînțelegem atât pachetul nivelului rețea cât și TPDU-ul încapsulat în el.

6.6.1 Probleme de performanță în rețelele de calculatoare

Unele probleme de performanță, cum este congestia, sunt cauzate de supraîncărcarea temporară a resurselor. Dacă apare subit o creștere de trafic la nivelul unui ruter peste nivelul care poate fi controlat de acesta, se va produce o congestie și performanțele vor avea de suferit. Congestia a fost studiată în detaliu în capitolul precedent.

Performanțele se degradează de asemenea în cazul unei dezechilibrări structurale a balanței resurselor. De exemplu, dacă o linie de comunicație de un gigabit este atașată la un terminal PC de performanță scăzută, procesorul slab nu va putea prelucra suficient de repede pachetele care sosesc, unele din acestea pierzându-se. Aceste pachete vor fi retransmise în cele din urmă, adăugând întârzieri, consumând din lărgimea de bandă și în general reducând performanțele.

Supraîncărcarea poate fi de asemenea inițiată în mod sincron. De exemplu, dacă un TPDU conține un parametru eronat (de exemplu portul pentru care este destinat), în multe cazuri receptorul va înapoia cu multă grijă un anunț de eroare. Să vedem acum ce s-ar putea întâmpla dacă un TPDU eronat ar fi răspândit către 10000 de mașini: fiecare din ele ar putea întoarce un mesaj de eroare. **Furtuna de difuzare** rezultată ar putea să scoată din funcțiune rețeaua. UDP a suferit de această problemă până când protocolul a fost modificat pentru a împiedica mașinile să răspundă erorilor cauzate de TPDU-urile UDP-ului trimise către adrese de difuzare.

Un al doilea exemplu de supraîncărcare sincronă este dat de efectele unei căderi a energiei electrice. Odată cu revenirea curentului, toate mașinile execută programul ROM de reinițializare. O secvență tipică de reinițializare poate cere în primul rând unui anume server (DHCP) identitatea precisă a mașinii, apoi poate cere unui server de fișiere o copie a sistemului de operare. Dacă sute de mașini execută acest lucru simultan, serverul va ceda probabil la această încărcare.

Chiar în absența unei supraîncărcări sincrone și chiar atunci când sunt suficiente resurse disponibile, performanțele pot fi slabe datorită unei proaste reglări a sistemului. De exemplu, dacă o mașină are suficientă memorie și putere de prelucrare, dar nu a fost alocat suficient spațiu pentru tampoane, vor apărea aglomerări și se vor pierde TPDU-uri. Similar, dacă algoritmul de planificare nu acordă o prioritate suficient de mare prelucrării TPDU-urilor care sunt recepționate, unele din ele se vor pierde.

Un alt element de reglare este potrivirea corectă a intervalelor de limită de timp. Atunci când este trimis un TPDU, în mod normal se poziționează un contor pentru a evita pierderea TPDU-lui. Dacă limita de timp este prea scurtă, se vor produce retransmisii inutile, obturând cablurile. Dacă limita de timp este prea lungă, se vor introduce întârzieri inutile după pierderea unui TPDU. Alți parametri ce pot fi reglați sunt lungimea intervalului de timp după care datele acumulate sunt confirmate, și numărul retransmisiilor făcute înainte de a se renunța.

Rețelele gigabit aduc cu ele noi probleme de performanță. Să considerăm, de exemplu, transmisia unei rafale de date de 64 K octeți de la San Diego la Boston, pentru a umple tamponul de 64K

octeți al receptorului. Să presupunem că legătura este de 1 Gbps și că întârzierea într-un singur sens prin fibra de sticlă este de 20 ms. Inițial, la momentul $t = 0$, conducta este goală, ca în fig. 6-41(a). Doar cu 500 μs mai târziu, în fig. 6-41(b), toate TPDU-urile sunt deja plasate pe fibră. Primul TPDU transmis va fi acum undeva în vecinătatea Brawley-ului, încă departe, în California de Sud. Cu toate acestea, transmisia trebuie să se oprească până se primește o actualizare de fereastră.

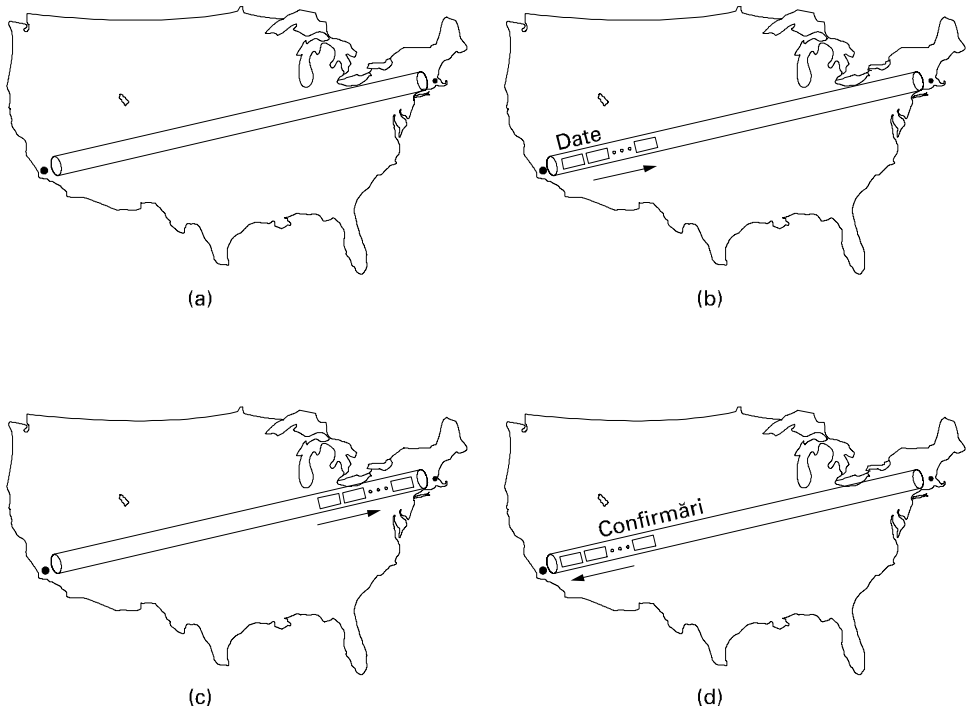


Fig. 6-41. Evoluția transmisiei unui megabit de la San Diego la Boston. (a) La momentul $t=0$. (b) După 500 μsec. (c) După 20 ms. (d) După 40 ms.

După 20 ms, primul TPDU atinge Boston-ul, ca în fig. 6-41(c), și este confirmat. În final, la 40 ms după momentul inițial, prima confirmare sosește înapoi la emițător și a doua rafală poate să fie transmisă. Cum linia de transmisie a fost utilizată pentru 0.5 ms din cele 40 ms, eficiența este undeva în jurul a 1.25 procente. Această situație este tipică pentru cazul folosirii protocoalelor vechi în linii gigabit.

O mărime utilă de reamintit când se analizează performanțele rețelei este produsul **lărgime de bandă-întârziere**. El este obținut prin înmulțirea lărgimii de bandă (în biți pe secundă) cu întârzierea traseului dus-întors (în secunde). Produsul reflectă capacitatea conductei de la emițător la receptor și înapoi (în biți).

De exemplu, în fig. 6-41 produsul lărgime de bandă-întârziere este de 40 milioane de biți. Cu alte cuvinte, pentru a funcționa la viteză maximă, emițătorul poate transmite rafale de 40 milioane de biți până la recepționarea primei confirmări. Umplerea conductei (în ambele sensuri) presupune o cantitate însemnată de biți. Acesta este motivul pentru care o rafală de jumătate de milion de biți atinge o eficiență de 1.25 procente: ea reprezintă doar 1.25 procente din capacitatea conductei.

Concluzia care poate fi trasă aici este că pentru atingerea unor performanțe bune, fereastra receptorului trebuie să fie cel puțin la fel de mare ca și produsul lărgime de bandă-întârziere, prefera-

bil chiar puțin mai mare, deoarece receptorul poate să nu răspundă instantaneu. Pentru o linie gigabit transcontinentală sunt necesari cel puțin 5 megabiți.

Dacă eficiența este dezastruoasă pentru un megabit, imaginați-vă cum ar arăta ea pentru o cerere scurtă de câteva sute de octeți. În cazul în care nu se găsește o altă utilitate pentru linie în timpul în care primul client este în așteptarea răspunsului, o linie gigabit nu este cu nimic mai bună ca o linie megabit, numai că este mult mai scumpă.

O altă problemă de performanță care poate apărea în aplicațiile critice din punctul de vedere al timpului, precum audio/video, este zgomotul. Un timp mediu de transmisie de valoare mică nu este suficient. Este necesară și o deviație standard mică. Obținerea atât a unui timp mediu de transmisie mic, cât și a unei deviații standard mici, cere un efort serios de inginerie.

6.6.2 Măsurarea performanțelor rețelei

Atunci când performanțele unei rețele sunt slabe, utilizatorii ei se plâng persoanelor care o administrează, cerând îmbunătățirea situației. Pentru a crește performanțele, operatorii trebuie mai întâi să determine cu exactitate ce se întâmplă. Pentru a afla ce se întâmplă în realitate, operatorii trebuie să efectueze măsurători. În această secțiune ne vom concentra asupra măsurării performanțelor rețelei. Discuția care urmează se bazează pe lucrarea lui Mogul (1993).

Ciclul de bază utilizat pentru îmbunătățirea performanțelor rețelei conține următorii pași:

1. Măsurarea performanțelor și a parametrilor relevanți ai rețelei.
2. Încercarea de a înțelege ceea ce se petrece.
3. Modificarea unuia din parametri.

Acești pași se repetă până la atingerea unor performanțe suficient de bune sau până când este clar că și ultima îmbunătățire posibilă a fost pusă în aplicare.

Măsurarea poate fi făcută în multe moduri și în multe locuri (atât fizic cât și în stiva de protocoale). Cel mai simplu mod de măsurare este inițializarea unui contor la începutul unei activități și observarea timpului necesar pentru îndeplinirea acelei sarcini. De exemplu, un element cheie în măsurare este aflarea timpului necesar unui TPDU pentru a fi confirmat. Alte măsurători sunt făcute cu contoare care înregistrează frecvența de apariție a unor evenimente (de exemplu, numărul de TPDU-uri pierdute). În final, unii pot fi interesați să afle valorile unor mărimi, ca de exemplu numărul de octeți prelucrați într-un anume interval de timp.

Măsurarea performanțelor și a parametrilor rețelei ascunde multe capcane potențiale. În cele ce urmează, enunțăm doar câteva din acestea. Orice încercare sistematică de a măsura performanțele rețelei trebuie să le evite.

Dimensiunea testului trebuie să fie suficient de mare

Timpul necesar pentru a trimite un TPDU nu trebuie măsurat o singură dată, ci în mod repetat, să zicem, de un milion de ori, luându-se în considerare media valorilor rezultate. Un test de dimensiune mare va reduce gradul de incertitudine în media și deviația standard a măsurătorii. Această incertitudine poate fi calculată pe baza formulelor statistice obișnuite.

Testele trebuie să fie reprezentative

Ideal ar fi ca întreaga secvență a celor un milion de măsurători să fie repetată în diferite momente ale zilei și ale săptămânii pentru a pune în evidență efectul diferențelor de încărcare a sistemului asupra mărimii măsurate. Măsurătorile de congestie, de exemplu, nu sunt prea utile dacă sunt făcute

într-un moment în care nu există nici o congestie. Uneori, rezultatele pot fi inițial contrare intuiției, de exemplu congestii importante la orele 10, 11, sau 1, 2 după amiază, dar nici un fel de congestie la amiază (când toți utilizatorii au pauză de prânz).

Utilizarea ceasurilor cu granularitate mare trebuie făcută cu atenție

Ceasurile calculatoarelor funcționează prin incrementarea la intervale regulate a unui anumit contor. De exemplu, un contor de milisecunde adaugă unu la contor la fiecare 1 ms. Utilizarea unui astfel de contor pentru a măsura un eveniment care durează mai puțin de 1 ms nu este imposibilă, dar necesită o oarecare atenție. (Desigur, unele calculatoare au ceasuri cu precizie mai bună).

Pentru a măsura intervalul necesar trimiterii unui TPDU, de exemplu, ceasul sistem (să spunem în milisecunde) ar trebui să fie citit în momentul în care se intră în codul de transport și din nou când se părăsește acest cod. Dacă timpul real de transmisie TPDU este de 300 μsec, diferența dintre cele două citiri va fi sau 0, sau 1, ambele valori greșite. Cu toate acestea, dacă măsurătoarea este repetată de un milion de ori și suma tuturor măsurătorilor este împărțită la un milion, timpul mediu va avea o precizie mai bună de 1 μsec.

Nu trebuie să se petreacă ceva neașteptat în timpul măsurătorilor

Efectuarea măsurătorilor pe un sistem universitar în ziua în care urmează să fie predat vreun proiect important poate conduce la rezultate diferite față de cele ce s-ar obține în ziua imediat următoare. Similar, dacă vreun cercetător se decide să organizeze o videoconferință în rețea, în timpul testelor, poate fi obținut un rezultat alterat. Cel mai bine este ca testele să fie rulate pe un sistem complet inactiv, întreaga sarcină fiind construită în vederea testării. Chiar și această abordare are propriile capcane. Deși ne-am aștepta ca nimeni să nu utilizeze rețeaua la ora 3 dimineața, acesta poate să fie chiar momentul în care un program de salvare automată începe să copieze conținutul tuturor discurilor pe bandă. Mai mult decât atât, s-ar putea să existe un trafic important pentru minunatele pagini de Web de pe rețeaua testată, trafic provenit din zone aflate pe alte meridiane orare.

Lucrul cu memoria ascunsă poate distruge măsurătorile

Modalitatea evidentă de a măsura timpul de transfer al fișierelor este de a deschide un fișier de dimensiune mare, de a-l citi în întregime și de a-l închide, urmând a vedea cât de mult a durat toată operația. Se repetă apoi operația de mult mai multe ori, pentru a obține o medie corectă. Problema este că sistemul poate memora fișierul în memoria ascunsă, astfel încât doar prima măsurătoare să implice traficul în rețea. Restul nu sunt decât accese la memoria ascunsă locală. Rezultatele unei astfel de măsurători sunt, în esență, fără nici o valoare (doar dacă nu cumva se dorește măsurarea performanțelor memoriei ascunse).

De obicei, se poate ocoli memoria ascunsă prin simpla ei supraîncărcare. De exemplu, dacă memoria ascunsă este de 10 megaocteți, ciclul de test ar putea deschide, citi și închide două fișiere de 10 megaocteți la fiecare buclă, în tentativa de a forța rata de succes în accesul la memoria ascunsă la 0. Cu toate acestea, dacă nu se înțelege cu absolută precizie algoritmul de manipulare a memoriei ascunse, trebuie procedat cu grijă.

Memorarea datelor în tampoane poate avea același efect. Se cunoaște un program utilitar popular pentru măsurarea performanțelor TCP/IP care raportează performanțe ale UDP-ului substanțial mai mari decât o permit liniile fizice. Cum se întâmplă acest lucru? Un apel către UDP întoarce în mod normal controlul odată ce mesajul a fost acceptat de către nucleu și adăugat la coada de transmisie. Dacă este suficient spațiu în tampon, măsurarea a 1000 de apeluri UDP nu înseamnă neapă-

rat că informațiile au fost transmise. Cea mai mare a informațiilor poate să se afle încă în nucleu, dar instrumentul de măsurare interpretează că ele au fost toate deja transmise.

Trebuie înțeles ceea ce se măsoară

Atunci când se măsoară timpul necesar pentru a citi un fișier de la distanță, măsurătorile depind de rețea, de sistemele de operare de la ambele capete - client și server, de tipul de echipament al plăcii de interfață utilizat, de programele care le controlează și de alți factori. Procedând cu atenție, putem determina în ultimă instanță timpul de transfer al fișierului pentru configurația utilizată. Dacă scopul îl reprezintă reglarea acestei configurații particulare, atunci măsurătorile sunt în regulă.

Cu toate acestea, dacă, în scopul alegerii unei interfețe de rețea pentru a fi cumpărată, sunt făcute măsurători similare pe trei sisteme diferite, rezultatele pot fi complet bulversate în cazul în care unul din programele care controlează echipamentul este de-a dreptul îngrozitor și utilizează doar 10% din performanțele plăcii.

Atenție la extrapolarea rezultatelor

Să presupunem că s-au făcut măsurători ale unei anumite mărimi prin simularea încărcării rețelei între 0 (sistem complet descărcat) și 0.4 (sistem încărcat în proporție de 40%), conform punctelor de date și liniilor continue dintre ele din fig. 6-42. Ar putea fi tentant să se extrapoleze liniar, așa cum o sugerează linia punctată. Cu toate acestea, multe din rezultatele anterioare indică un factor de $1/(1 - \rho)$, unde ρ este încărcarea, astfel încât valorile adevărate pot arăta mai degrabă ca linia întreruptă, care crește mult mai repede decât liniar.

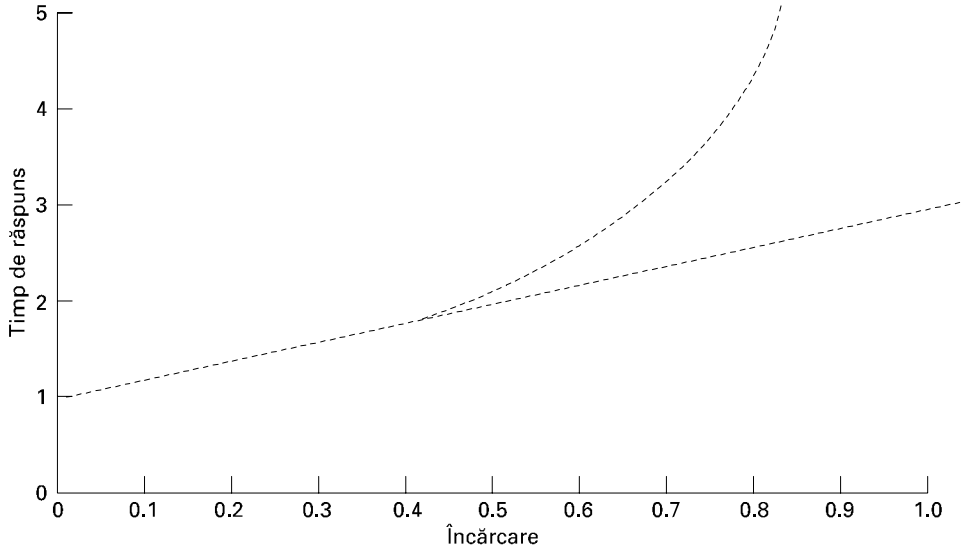


Fig. 6-42. Răspunsul, ca o funcție de încărcare.

6.6.3 Proiectarea de sistem pentru performanțe superioare

Măsurarea și ajustarea pot îmbunătăți considerabil performanțele, dar ele nu pot substitui o proiectare făcută bine de la început. O rețea proiectată superficial poate fi îmbunătățită tot în aceeași măsură. O soluție mai bună este să se refacă totul de la temelie.

În această secțiune, vom prezenta câteva reguli rezultate dintr-o experiență acumulată pe un număr mare de rețele. Aceste reguli se referă nu doar la proiectarea de rețea, ci și la proiectarea de sistem, și aceasta pentru că programul și sistemul de operare sunt deseori mai importante decât ruterul și echipamentul de interfață. Cele mai multe din aceste idei constituiau cunoștințe de bază ale proiectanților de rețele, care erau propagate din generație în generație pe cale orală. Ele au fost prima oară enunțate explicit de Mogul (1993); expunerea noastră merge în mare parte în paralel cu a sa. O altă sursă relevantă este (Metcalfe, 1993).

Regula #1: Viteza procesorului este mai importantă decât viteza rețelei.

O experiență îndelungată a arătat că în aproape toate rețelele, supraîncărcarea indusă de sistemul de operare și de protocol domină de fapt timpul de transmisie pe cablu. De exemplu, în teorie, timpul minim pentru un RPC pe o rețea Ethernet este de 102 μs, corespunzând unei cereri minime (64 de octeți) urmate de un răspuns minim (64 de octeți). În realitate, evitarea întârzierii suplimentare introduse de software și obținerea timpului RPC cât de cât apropiat de cel teoretic este o realizare considerabilă.

Similar, cea mai mare problemă în funcționarea la 1 Gbps constă în plasarea biților din tamponul utilizatorului pe fibră suficient de repede și în recepționarea acestor biți de către procesorul receptor la fel de repede cum sosesc. Pe scurt, dacă se dublează viteza procesorului, de regulă se poate ajunge până aproape de dublarea productivității. Dublarea capacității rețelei nu are de regulă nici un efect, deoarece gâtuirea se produce în general la calculatoarele gazdă.

Regula #2: Reducerea numărului de pachete pentru a reduce supraîncărcarea datorată programelor.

Prelucrarea unui TPDU adaugă o anumită supraîncărcare per TPDU (de exemplu prelucrarea antetului) și o anumită cantitate de prelucrare suplimentară per octet (de exemplu calculul sumei de control). Atunci când se trimit un milion de octeți, supraîncărcarea per octet este aceeași, indiferent dacă dimensiunea TPDU variază. Cu toate acestea, utilizarea de TPDU-uri de 128 octeți presupune de 32 de ori mai multă supraîncărcare per TPDU față de cazul a 4K octeți per TPDU. Această supraîncărcare crește cu rapiditate.

În plus față de supraîncărcarea TPDU-ului, trebuie considerată și supraîncărcarea datorată nivelurilor inferioare. Fiecare sosire a unui pachet generează o întrerupere. Pe un procesor cu bandă de asamblare modern, fiecare întrerupere fragmentează banda de asamblare a procesorului, interferă cu memoria tampon, presupune o schimbare de context în controlul memoriei și impune salvarea unui număr important din registrele procesorului. O divizare prin n a numărului de TPDU-uri trimise reduce în consecință numărul de întreruperi și supraîncărcarea pachetelor cu un factor de n.

Această observație justifică necesitatea colectării unei cantități importante de date înaintea transmisiei, în scopul reducerii numărului de întreruperi la celălalt capăt. Algoritmul Nagle și soluția Clark pentru sindromul ferestrei stupide reprezintă încercări de a face exact acest lucru.

Regula #3: Minimizarea numărului de comutări de context.

Comutările de context (de exemplu, din mod nucleu în mod utilizator) sunt catastrofale. Ele au aceleași proprietăți incomode ca și întreruperile, cea mai rea fiind o lungă serie de eșecuri la accesul inițial la memoria ascunsă. Comutările de context pot fi reduse dacă funcția de bibliotecă ce trimite date le stochează intern până la acumularea unei cantități semnificative. Similar, de partea receptorului, TPDU-urile de dimensiune mică recepționate ar trebui colectate și trimise utilizatorului dintr-un singur foc și nu individual, în scopul minimizării comutărilor de context.

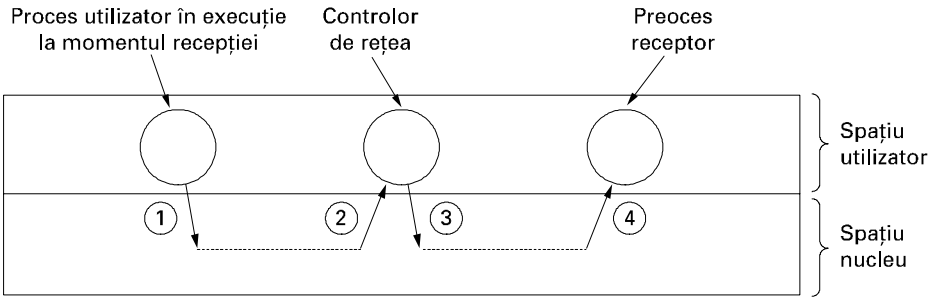


Fig. 6-43. Patru comutări de context pentru fiecare pachet, cu un controlor de rețea în spațiul utilizator.

În cel mai bun caz, sosirea unui pachet produce o comutare de context din modul utilizator curent în modul nucleu și apoi o comutare către procesul receptor, astfel încât acesta din urmă să preia informația nou sosită. Din nefericire, în multe sisteme de operare se petrec comutări de contexte suplimentare. De exemplu, dacă procesul controlor de rețea rulează ca un proces special în spațiul utilizator, sosirea unui pachet provoacă, probabil, o comutare de context de la utilizatorul curent către nucleu, apoi încă o comutare de la nucleu către controlorul de rețea, urmată de încă una înapoi către nucleu și în final din nucleu către procesul receptor. Această secvență este prezentată de fig. 6-43. Toate aceste comutări de contexte pentru fiecare pachet consumă mult din timpul procesorului și au un efect distrugător asupra performanțelor rețelei.

Regula #4: Minimizarea numărului de copieri.

Efectuarea unor copii multiple este și mai dăunătoare decât comutările multiple de contexte. Nu este nimic deosebit în faptul că un pachet proaspăt recepționat este copiat de trei sau patru ori înainte ca TPDU-ul conținut în el să fie livrat. După ce un pachet este recepționat de către echipamentul de rețea într-un tampon special cablat pe placă, el este copiat de obicei într-un tampon al nucleului. De aici el este copiat într-un tampon al nivelului rețea, apoi într-unul al nivelului transport și în final de către procesul aplicației receptoare.

Un sistem de operare inteligent va copia câte un cuvânt odată, dar nu este deloc neobișnuit să fie necesare cinci instrucțiuni per cuvânt (încărcare, memorare, incrementarea unui registru index, un test pentru marcajul de sfârșit al datelor și un salt condiționat). A face trei copii ale fiecărui pachet la cinci instrucțiuni copiate pe fiecare cuvânt de 32 de biți, necesită 15/4 sau aproximativ patru instrucțiuni per octet copiat. Pe un procesor de 500 MIPS, o instrucțiune durează 2 ns deci fiecare octet necesită 8 ns sau aproximativ 1 ns per bit, dând o rată de transfer de aproximativ 1 Gbps. Ținând cont și de prelucrarea antetului, tratarea întreruperilor și a comutărilor de contexte, se pot atinge 500 Mbps, aceasta fără a pune la socoteală prelucrarea efectivă a datelor. Evident, controlul unei linii Ethernet de 10 Gbps care funcționează la viteză maximă, nici nu intră în discuție.

De fapt, probabil că nici o linie de 500 Mbps nu poate fi manipulată la viteză maximă. În calculul anterior am presupus că o mașină cu 500 MIPS poate executa oricare 500 milioane de instrucțiuni pe secundă. De fapt, mașinile pot opera la această viteză doar dacă nu execută referiri la memorie. Operațiile cu memoria sunt, de cele mai multe ori, de zece ori mai lente decât instrucțiunile registru-registru (adică 20 ns / instrucțiune). Dacă 20 de procente din instrucțiuni chiar referă memoria (adică datele necesare nu se află în memoria tampon), ceea ce este de așteptat când este vorba despre pachetele care vin, timpul mediu de execuție este 5.6 ns (0.8x2+0.2x20). Cu patru instrucțiuni/octet,

avem nevoie de 22.4 ns / octet, sau 2.8 ns / bit, care înseamnă aproximativ 357 Mbps. Considerând un procent de 50 supraîncărcare obținem 178 Mbps. De notat că echipamentul suport nu ajută în acest caz. Problema este că sistemul de operare execută prea multe operații de copiere.

Regula #5: Oricând se poate cumpăra mai multă lărgime de bandă, dar niciodată o întârziere mai mică.

Următoarele trei reguli se ocupă de comunicație mai mult decât de prelucrarea protocolului. Prima regulă stabilește că, dacă se dorește o lățime de bandă mai mare, este suficient să o cumperi. Dacă se pune o a doua fibră alături de prima, se dublează lărgimea de bandă, dar nu se micșorează deloc întârzierile. Micșorarea întârzierilor presupune îmbunătățirea programului de protocol, a sistemului de operare sau a interfeței cu rețeaua. Chiar dacă toate acestea sunt îndeplinite, întârzierea nu se va reduce dacă gâtuirea constă în timpul de transmisie.

Regula #6: Evitarea congestiei este preferabilă eliminării congestiei.

Vechea maximă conform căreia o uncie de prevenire este mai bună decât o livră de însănătoșire este cu certitudine valabilă și în cazul congestiei rețelei. Atunci când o rețea este congestionată, se pierd pachete, lărgimea de bandă este irosită, apar întârzieri inutile și multe altele. Recuperarea din congestie ia timp și răbdare. Este de preferat să se procedeze de așa natură, încât congestia să nu apară. Evitarea congestiei este ca și vaccinarea împotriva tetanosului: doare puțin în momentul în care se face vaccinul, dar aceasta împiedică o durere mult mai mare mai târziu.

Regula #7: Evitarea întârzierilor.

În rețele sunt necesare contoare, dar ele ar trebui utilizate cu măsură și întârzierile ar trebui minimizate. Atunci când expiră un contor, se repetă de regulă o acțiune. Dacă este într-adevăr necesară repetarea acțiunii respective, atunci asta este, dar repetarea ei fără rost reprezintă o risipă.

Modalitatea de a evita un efort suplimentar constă în înțelegerea faptului că aceste contoare se cam situează de partea conservatoare a lucrurilor. Un contor căruia îi ia mult timp ca să expire adaugă o mică întârziere suplimentară în cazul (puțin probabil) în care un TPDU se pierde. Un contor care expiră atunci când nu ar trebui, consumă în mod nepermis din timpul procesor, irosește din lărgimea de bandă și adaugă o încărcare suplimentară în zeci de rutere, probabil fără nici un motiv serios.

6.6.4 Prelucrarea rapidă a TPDU-urilor

Morala poveștii anterioare este aceea că principalul obstacol către rețelele rapide îl constituie programul de protocol. În această secțiune, vom studia câteva modalități pentru a crește viteza acestui program. Pentru mai multe informații, pot fi citite (Clark și alții 1989; și Chase și alții, 2001).

Supraîncărcarea indusă de prelucrarea TPDU-urilor are două componente: supraîncărcare per TPDU și supraîncărcare per octet. Ambele trebuie să fie abordate. Cheia accelerării prelucrării TPDU-urilor constă în separarea cazului normal (transferul datelor într-un singur sens) și tratarea sa specială. Cu toate că este necesară o secvență de TPDU-uri speciale pentru a se intra într-o stare *STABILIT*, odată intrat în starea respectivă, prelucrarea TPDU-urilor decurge lin, până în clipa în care una din părți inițiază închiderea conexiunii.

Să începem examinarea părții emițătoare din starea *STABILIT*, atunci când există date de transmis. Pentru claritate, presupunem că entitatea transport este în nucleu, aceleași idei aplicându-se și în cazul în care este vorba de un proces în spațiul utilizator sau o bibliotecă în interiorul proce-

sului emițător. În fig. 6-44, pentru a executa SEND-ul, procesul emițător intră în mod nucleu prin acționarea unei capcane. Primul lucru pe care îl face entitatea transport este de a testa dacă nu cumva este vorba de cazul normal: starea este *STABILIT*, nici o parte nu încearcă să închidă conexiunea, un TPDU normal (adică unul care nu e în afara limitelor) este în curs de a fi transmis și la receptor este disponibil un spațiu fereastră suficient. Dacă toate condițiile sunt îndeplinite, nici un test suplimentar nu mai este necesar și poate fi acaparată calea rapidă prin entitatea de transport emițătoare. De obicei, această cale este acaparată în majoritatea timpului.

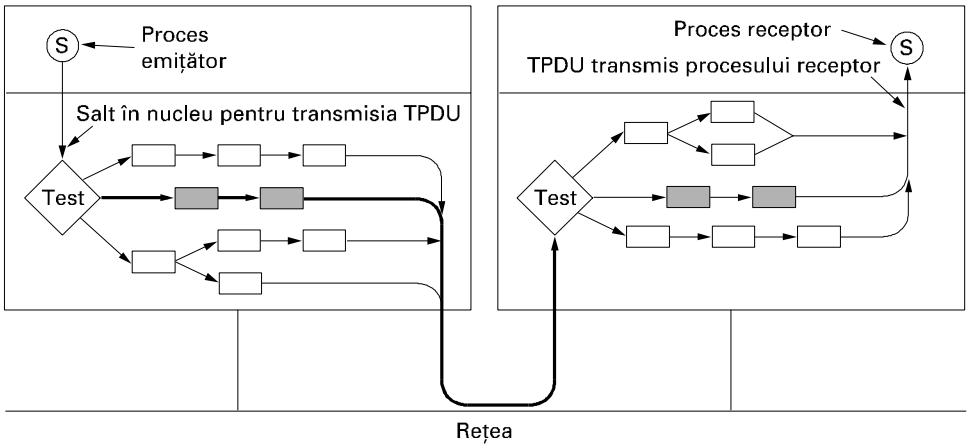


Fig. 6-44. Calea rapidă de la emițător la receptor este indicată printr-o linie groasă. Pașii de prelucrare ai acestei căi sunt reprezentați prin dreptunghiuri umbrite.

În cazul obișnuit, antetele mai multor date TPDU consecutive sunt în mare parte identice. Pentru a profita de acest lucru, în interiorul entității transport se memorează un antet prototip. La începutul căii rapide, el este copiat cât de repede posibil într-un tampon special, cuvânt cu cuvânt. Acele câmpuri care ulterior se modifică de la un TPDU la altul sunt suprascrise în tampon. În mod frecvent, aceste câmpuri sunt ușor de derivat din variabilele de stare, de exemplu următorul număr de secvență. Apoi este pasat nivelului rețea un indicator spre întregul antet TPDU, împreună cu un indicator spre informația utilizator. Și aici se poate urma aceeași strategie (caz neacoperit de fig. 6-44). În final, nivelul rețea furnizează pachetul rezultat nivelului legătură de date, în vederea transmisiei.

Pentru a exemplifica modul în care operează acest principiu în practică, să considerăm cazul TCP/IP-ului. Fig. 6-45(a) prezintă antetul TCP. Câmpurile hașurate sunt identice între două TPDU-uri consecutive, pe un flux într-un singur sens. Tot ce are de făcut entitatea transport este să copieze cele cinci cuvinte dintr-un antet prototip în tamponul care urmează să fie transmis, să completeze următorul număr de secvență (prin copierea lui dintr-un cuvânt din memorie), să calculeze suma de control și să incrementeze numărul de secvență din memorie. Entitatea poate înmâna apoi antetul, împreună cu datele aferente, unei proceduri IP speciale, în vederea transmisiei obișnuite a unui TPDU de dimensiune maximă. În continuare IP copiază cele cinci cuvinte ale antetului său prototip [vezi fig. 6-45(b)] în tampon, completează câmpul *Identificare* și calculează suma sa de control. Pachetul este acum gata pentru transmisie.

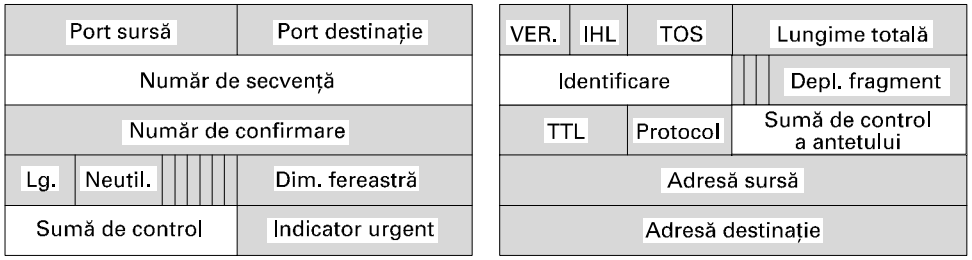


Fig. 6-45. (a) Antetul TCP. (b) Antetul IP. În ambele cazuri, câmpurile haşurate se obțin din prototip, fără nici o modificare.

Să aruncăm o privire asupra prelucrării pe calea rapidă în cazul receptorului din fig. 6-44. Pasul 1 constă în localizarea înregistrării de conexiune din TPDU-ul recepționat. În cazul TCP, înregistrarea de conexiune poate fi memorată într-o tabelă de dispersie pentru care cheia poate fi o funcție simplă aplicată celor două adrese IP și celor două porturi. Odată ce înregistrarea de conexiune a fost localizată, corectitudinea sa trebuie verificată prin compararea ambelor adrese și ambelor porturi.

O optimizare care accelerează și mai mult determinarea înregistrării de conexiune constă în menținerea unui indicator către ultima înregistrare utilizată, urmând ca aceasta să fie prima înregistrare testată. Clark ș.a. (1989) au aplicat această idee și au observat o rată de succes care depășește 90%. Alte euristici de căutare sunt descrise în (McKenney și Dove, 1992).

În continuare, TPDU-ul este verificat pentru a determina dacă este vorba de cazul normal: starea este *STABILIT*, niciuna din părți nu încearcă închiderea conexiunii, este un TPDU complet, nici un indicator special nu este poziționat și numărul de secvență este cel așteptat. Aceste teste înseamnă doar câteva instrucțiuni. Dacă toate condițiile sunt îndeplinite, este invocată o procedură TCP pentru cale rapidă.

Calea rapidă actualizează înregistrarea de conexiune și copiază informația către utilizator. Totodată, suma de control este calculată chiar pe parcursul copierii, eliminând astfel trecerile suplimentare pe secvența de date. Dacă suma de control este corectă, înregistrarea de conexiune este actualizată și se trimite o confirmare. Schema generală care constă într-un control rapid la început, pentru a vedea dacă antetul este cel așteptat, precum și în existența unei proceduri speciale care tratează cazul respectiv, se numește **predicția antetului**. Schema este utilizată în multe din implementările TCP. Atunci când această optimizare este utilizată împreună cu celelalte optimizări discutate în acest capitol, este posibil ca TCP-ul să atingă la execuție 90% din viteza de copiere locală memorie - memorie, presupunând că mediul de comunicație este suficient de rapid.

Alte două domenii unde se pot obține câștiguri importante în performanțe sunt controlul tampoanelor și al contoarelor de timp. În controlul tampoanelor, ideea constă în evitarea copierilor inutile, așa cum s-a menționat anterior. Controlul contoarelor de timp este important, deoarece aproape nici un contor nu expiră de fapt. Acestea sunt poziționate astfel, încât să ne păzească împotriva pierderii de TPDU-uri, dar majoritatea TPDU-urilor, ca și confirmările lor, de altfel, ajung corect la destinație. Este deci important să se optimizeze controlul contoarelor de timp pentru cazul în care acestea expiră rar.

O schemă uzuală constă în utilizarea unei liste îmbunătățite a evenimentelor generate de contoare, sortate în funcție de momentul expirării acestora. Intrarea din capătul acestei liste conține un contor care indică depărtarea în impulsuri de ceas față de momentul expirării. Fiecare din intrările

care urmează indică printr-un contor depărtarea în impulsuri de ceas față de intrarea precedentă. Astfel, pentru evenimente care expiră în 3, 10 și respectiv 12 impulsuri, cele trei contoare sunt 3, 7 și, respectiv, 2.

La fiecare impuls de ceas, contorul din capătul listei este decrementat. Atunci când contorul atinge valoarea 0, se prelucrează evenimentul asociat lui și următorul element din listă devine capătul listei. Contorul acestuia nu trebuie să fie modificat. Prin această schemă, inserarea și ștergerea evenimentelor sunt operații costisitoare, cu un timp de execuție proporțional cu lungimea listei.

Dacă intervalul maxim de expirare al contorului este limitat și cunoscut în avans, poate fi utilizată o abordare mai eficientă. În acest caz, poate fi utilizat un vector numit **roata timpului**, ca în fig. 6-46. Fiecare poziție corespunde unui impuls de ceas. Ceasul curent reprezentat este $T = 4$. Contoarele sunt planificate să expire la 3, 10 și 12 impulsuri față de acest moment. Dacă un contor nou este brusc poziționat să expire peste 7 impulsuri, se creează pur și simplu o intrare în poziția 11. Similar, în cazul în care contorul planificat să expire la $T + 10$ trebuie să fie anulat, trebuie parcursă lista care începe pe poziția 14 și intrarea cerută trebuie ștearsă. Să observăm că vectorul din fig. 6-46 nu poate suporta contoare care expiră după $T + 15$.

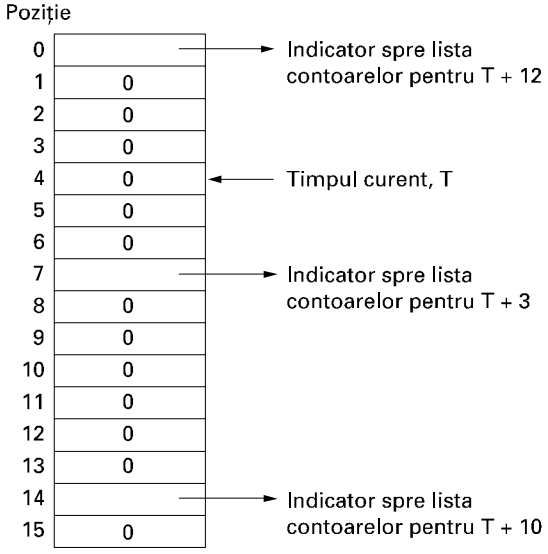


Fig. 6-46. O roată a timpului.

Cu fiecare impuls de ceas indicatorul de timp curent este avansat cu o poziție (circular). Dacă intrarea indicată este nenulă, sunt prelucrate toate contoarele asociate ei. Mai multe variațiuni pe această temă sunt discutate în (Varghese și Lauck, 1987).

6.6.5 Protocoale pentru rețele gigabit

La începutul anilor '90 au început să apară rețele gigabit. Prima reacția a oamenilor a fost să utilizeze pentru ele vechile protocoale, lucru care a pus în scurt timp diferite probleme. Vom discuta în această secțiune câteva din aceste probleme, precum și direcțiile urmate de noile protocoale pentru a le soluționa, pe măsură ce evoluăm către rețele tot mai rapide.

Prima problemă este că multe protocoale utilizează secvențe de numere de 32 de biți. Când a apărut Internetul, liniile între rutere erau mai ales linii închiriate de 56 Kbps, astfel că unui calculator gazdă care mergea la viteză maximă îi trebuia mai mult de o săptămână să treacă prin toate numerele de secvență. Pentru proiectanții de TCP, 2^{32} a fost o aproximație destul de decentă a infinitului, pentru că era mic pericolul ca pachete vechi să mai fie prin preajmă la o săptămână după ce au fost transmise. Cu o linie Ethernet de 10 Mbps, timpul de parcurgere a numerelor de secvență a devenit de 57 de minute, mult mai mic, dar încă administrabil. Cu o linie Ethernet de 1 Gbps transmițând date prin Internet, timpul de parcurgere este de aproximativ 34 secunde, mult sub timpul maxim de viață al unui pachet în Internet, de 120 de secunde. Dintr-o dată, 2^{32} nu mai este o aproximare atât de bună a infinitului, din moment ce un transmițător poate cicla prin spațiul de secvență, în timp ce pachetele vechi există încă. RFC 1323 oferă totuși o fereastră de ieșire.

Problema este că mulți proiectanți de protocoale au plecat de la presupunerea că timpul necesar consumării spațiului numerelor de secvență depășește cu mult timpul maxim de viață al unui pachet. În consecință, nu era nici măcar nevoie să își facă griji că duplicate vechi pot să existe încă undeva atunci când numerele de secvență au revenit la vechile valori. La viteze gigabit această presupunere cade.

O a doua problemă este aceea că viteza de comunicație a crescut mult mai repede decât viteza de prelucrare. (Notă pentru inginerii de calculatoare: Ieșiți în stradă și bateți-i pe inginerii de comunicații! Ne bazăm pe voi.) În anii '70, ARPANET-ul opera la 56 Kbps și avea calculatoare care funcționau la aproape 1 MIPS. Pachetele erau de 1008 biți și astfel ARPANET-ul putea livra aproximativ 56 pachete/sec. Având disponibile 18 ms pentru fiecare pachet, o mașină gazdă putea să-și permită să irosească 18000 instrucțiuni pentru prelucrarea unui pachet. Desigur, dacă ar fi făcut astfel, ar fi asfixiat complet procesorul, dar putea renunța la doar 9000 instrucțiuni per pachet și tot i-ar mai fi rămas jumătate din puterea procesorului pentru celelalte prelucrări.

Să comparăm aceste numere cu calculatoarele moderne de 1000 MIPS care interschimbă pachete de 1500 de octeți pe o linie gigabit. Pachetele pot curge cu o viteză de peste 80000 pe secundă, astfel încât, dacă vrem să rezervăm jumătate din puterea procesorului pentru aplicații, prelucrarea unui pachet trebuie să se încheie în 6,25 μsec. În 6,25 μsec, un calculator de 1000 MIPS poate executa doar 6250 instrucțiuni, doar 1/3 din ceea ce își putea permite un calculator gazdă ARPANET. Mai mult decât atât, instrucțiunile RISC moderne fac mai puține lucruri per instrucțiune decât o făceau vechile instrucțiuni CISC, deci problema este chiar mai gravă decât pare. În concluzie: există mult mai puțin timp pentru prelucrarea efectuată de protocol decât exista altădată, deci protocoalele trebuie să devină mai simple.

O a treia problemă este aceea că protocolul cu întoarcere în *n* pași are performanțe slabe pe linii cu un produs lărgime de bandă-întârziere de valoare mare. Să considerăm de exemplu o linie de 4000 km operând la 1 Gbps. Timpul de transmisie dus-întors este de 40 ms, timp în care un emițător poate transmite 5 megaocteți. Dacă se detectează o eroare, atunci vor fi necesare 40 ms înainte ca emițătorul să fie avertizat de acest lucru. Dacă este utilizat algoritmul cu întoarcere în *n* pași, emițătorul nu va avea de retransmis doar pachetul eronat, ci și toți cei 5 megaocteți care au urmat după el. În mod clar, are loc o risipă mare de resurse.

O a patra problemă este aceea că liniile gigabit sunt fundamental diferite de liniile megabit, liniile gigabit de lungime mare fiind limitate de întârziere mai degrabă decât de lărgimea de bandă. În fig. 6-47 arătăm timpul necesar transferului unui fișier de un megabit la 4000 km distanță pentru diferite viteze de transfer. La o viteză de până la 1 Mbps, timpul de transmisie este dominat de viteza la care pot fi transferați biții. La 1 Gbps, întârzierea de 40 ms a circuitului dus-întors domină acea

milisecundă necesară pentru a pune bitul pe fir. Creșteri suplimentare ale lărgimii de bandă aproape că rămân fără efect.

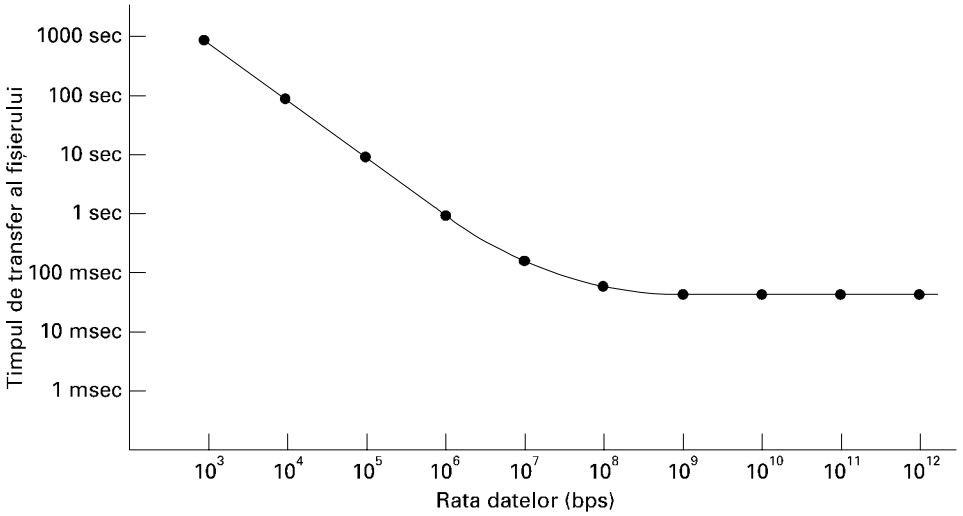


Fig. 6-47. Timpul necesar transferului și confirmării unui fișier de un megabit pe o linie de 4000 km.

Fig. 6-47 are implicații nedorite pentru protocoalele de rețea. Ea arată că protocoalele pas-cu-pas (stop-and-wait), precum RPC, au o limită superioară inerentă de performanță. Această limită este dictată de viteza luminii. Oricât de mare ar fi progresul tehnologic în optică, nu se poate obține nici o îmbunătățire (deși noi legi ale fizicii ar fi folositoare).

O a cincea problemă care merită menționată nu este o problemă de protocol sau o problemă tehnologică, ci un rezultat al noilor aplicații. Enunțată doar, ea spune că pentru multe aplicații gigabit, precum multimedia, varianța sosirii pachetelor în timp este la fel de importantă ca însăși media întârzierilor. De cele mai multe ori este de preferat o viteză de livrare redusă, dar uniformă uneia rapide, dar fluctuante.

Să revenim acum de la identificarea problemelor la modalitățile lor de rezolvare. Vom face, pentru început, câteva remarci generale, apoi vom analiza mecanismele de protocol, aspectul pachetelor și programele de protocol.

Principiul de bază pe care toți proiectanții de rețele gigabit ar trebui să-l învețe pe de rost este:

Proiectați astfel încât să optimizați viteza, nu lățimea de bandă.

Protocoalele vechi au fost de regulă proiectate pentru a minimiza numărul de biți de pe fir, prin utilizarea frecventă a câmpurilor de dimensiuni mici și prin împachetarea lor în octeți și cuvinte. În zilele noastre, există suficient de multă lățime de bandă. Problema o reprezintă prelucrările efectuate de protocoale, deci acestea trebuie reduse la minim. Clar că proiectanții procesorului IPv6 au înțeles acest principiu.

O modalitate tentantă pentru accelerarea lucrurilor este de a construi interfețe rapide cu rețeaua sub formă de echipamente fizice. Dificultatea acestei strategii constă în aceea că, în lipsa unui protocol extrem de simplu, un nou echipament înseamnă conectarea unei noi plăci, cu un al doilea

procesor și cu propriul program. Pentru a asigura faptul că un co-procesor de rețea este mai ieftin decât procesorul principal, el este de regulă o componentă mai lentă. Consecința acestei soluții este că mult timp, procesorul principal (rapid) este inactiv, așteptând ca al doilea procesor (cel lent) să facă toată munca critică. Este un mit acela că procesorul principal ar avea altceva de făcut în acest timp. Mai mult decât atât, atunci când comunică două procesoare independente, pot apărea condiții de cursă, fiind deci necesare protocoale complexe, pentru sincronizarea lor corectă. În general, cea mai bună abordare este de a concepe protocoalele cât mai simplu și de a lăsa procesorul principal să facă toată treaba.

Să studiem acum noțiunea de reacție în protocoalele de mare viteză. Datorită buclei de întârziere (relativ) lungă, reacția ar trebui evitată: receptorului îi ia prea mult timp pentru a anunța emițătorul. Un exemplu de reacție este controlul vitezei de transfer prin utilizarea unui protocol cu fereastră glisantă. Pentru a evita întârzierile (îndelungate) inerente atunci când receptorul trimite emițătorului actualizările de fereastră, cel mai bine este să se utilizeze o rată de transfer dictată de protocol. Într-un astfel de protocol, emițătorul poate trimite tot ceea ce dorește el să trimită, cu condiția să nu o facă mai repede decât s-a convenit inițial cu receptorul.

Un al doilea exemplu de reacție este algoritmul startului lent al lui Jacobson. Acest algoritm face interogări multiple pentru a determina cât de mult poate suporta rețeaua. În rețelele de mare viteză utilizarea a jumătate de duzină de interogări scurte pentru testarea răspunsul rețelei conduce la irosirea unei părți imense din lățimea de bandă. O schemă mai eficientă este ca emițătorul, receptorul și rețeaua să rezerve resursele necesare în momentul stabilirii conexiunii. Rezervarea în avans a resurselor are de asemenea avantajul diminuării fluctuațiilor. Pe scurt, migrarea spre viteze mari împinge inexorabil proiectarea spre operații orientate pe conexiuni sau spre ceva foarte apropiat de acestea. Desigur, dacă lărgimea de bandă nu va mai fi o problemă în viitor încât nimănui să nu-i mai pese de pierderi, regulile de proiectare vor deveni foarte diferite.

În rețelele gigabit, o importanță deosebită trebuie acordată organizării pachetelor. Antetul ar trebui să conțină cât mai puține câmpuri cu putință pentru a reduce timpul de prelucrare, iar aceste câmpuri ar trebui să fie suficient de mari pentru a-și îndeplini scopul și pentru a fi aliniate la cuvânt, fiind astfel mai ușor de prelucrat. În acest context, „suficient de mari” înseamnă eliminarea problemelor precum revenirea numerelor de secvență la valori vechi cât timp există încă pachete vechi, incapacitatea receptorilor de a oferi suficient spațiu de fereastră datorită unei dimensiuni prea mici a câmpului fereastră ș.a.m.d.

Antetul și informația ar trebui să fie prevăzute cu sume de control separate, din două motive. În primul rând, pentru a face posibilă calcularea sumei de control a antetului, dar nu și a datelor. În al doilea rând, pentru a determina corectitudinea antetului înaintea începerii copierii datelor în spațiul utilizator. Este de dorit să se calculeze suma de control a datelor la momentul copierii lor în spațiul utilizator, dar dacă antetul este incorect, copierea poate să se facă în spațiul unui alt proces. Pentru a evita o copiere incorectă, dar pentru a permite și calculul sumei de control a datelor în timpul copierii, este esențial ca cele două sume de control să fie separate.

Dimensiunea maximă a datelor ar trebui să fie mare pentru a permite operații eficiente chiar și în situația transferului la distanțe mari. De asemenea, cu cât este mai mare blocul de informație, cu atât este mai mică fracțiunea din totalitatea lățimii de bandă alocată antetului. 1500 de octeți este prea puțin.

O altă caracteristică importantă este posibilitatea de a trimite o cantitate rezonabilă de informație o dată cu cererea de conexiune. În acest fel se poate salva un timp de comunicație dus-întors.

În sfârșit, sunt utile câteva cuvinte despre programul de protocol. Cea mai mare atenție trebuie acordată cazului de succes. Multe din protocoalele vechi aveau tendința de a evidenția ce este de făcut atunci când ceva nu mergea cum trebuie (de exemplu pierderea unui pachet). Pentru a face protocoalele să meargă mai repede, proiectanții ar trebui să aibă ca scop minimizarea timpului de prelucrare atunci când totul funcționează corect. Minimizarea timpului de prelucrare în caz de eroare trebuie să treacă pe planul doi.

Un al doilea aspect legat de programe este minimizarea timpului de copiere. Așa cum am văzut mai devreme, copierea datelor introduce în general un cost suplimentar. Ideal ar fi ca echipamentul fizic să depoziteze în memorie fiecare pachet recepționat ca un bloc contiguu de date. Programul ar trebui apoi să copieze acest pachet în tamponul utilizator printr-o singură copiere de bloc. În funcție de modul de lucru al memoriei tampon, ar fi de dorit chiar să se evite copierea în buclă. Cu alte cuvinte, cel mai rapid mod de a copia 1024 de cuvinte este de a avea 1024 de instrucțiuni MOVE una după cealaltă (sau 1024 de perechi încărcare-memorare). Rutina de copiere este critică într-o asemenea măsură, încât, dacă nu există altă modalitate de a păcăli compilatorul ca să producă cu precizie codul optimal, ea ar trebui scrisă de mână, cu multă atenție, direct în cod de asamblare.

6.7 REZUMAT

Nivelul transport reprezintă cheia pentru înțelegerea protocoalelor organizate pe niveluri. El furnizează diferite servicii, cel mai important dintre acestea fiind fluxul de octeți capăt-la-capăt de la emițător la receptor, fiabil și orientat pe conexiuni. El este accesat prin primitive de serviciu care permit stabilirea, utilizarea și eliberarea conexiunilor. O interfață de nivel de transport obișnuită este cea oferită de soclurile Berkeley.

Protocoalele de transport trebuie să fie capabile să controleze conexiunea în rețele nefiabile. Stabilirea conexiunii este complicată de existența pachetelor duplicate întârziate, care pot apărea la momente inoportune. Pentru a le face față, stabilirea conexiunii trebuie făcută prin intermediul protocoalelor cu înțelegere în trei pași. Eliberarea unei conexiuni este mai simplă decât stabilirea sa, dar este încă departe de a fi banală datorită problemei celor două armate.

Chiar și în cazul unui nivel rețea complet fiabil, nivelul transport are suficient de mult de lucru. El trebuie să controleze toate primitivele de serviciu, toate conexiunile și contoarele de timp și trebuie să aloce și să utilizeze credite.

Internetul are două protocoale de transport principale: UDP și TCP. UDP este un protocol neorientat pe conexiune care este în principal un ambalaj pentru pachetele IP, cu caracteristicile suplimentare de multiplexare și demultiplexare a proceselor multiple folosind o singura adresa de IP. UDP poate fi folosit pentru interacțiunile client-server, de exemplu, utilizând RPC. De asemenea poate fi folosit pentru construirea protocoalelor în timp real cum ar fi RTP.

Principalul protocol de transport în Internet este TCP. El oferă un flux sigur, bidirecțional de octeți. El utilizează un antet de 20 de octeți pentru toate segmentele. Segmentele pot fi fragmentate de rutere în cadrul Internet-ului, deci calculatoarele gazdă trebuie să fie pregătite să le reasambleze. S-a depus un mare efort pentru optimizarea performanțelor TCP-ului, utilizând algoritmii propuși de Nagle, Clark, Jacobson, Karn și alții. Legăturile fără fir adaugă o varietate de complicații TCP-ului.

TCP Tranzacțional este o extensie a TCP-ului care se ocupă de interacțiunile client-server cu un număr redus de pachete.

Performanțele rețelei sunt dominate în mod tipic de protocol și de costul suplimentar datorat tratării TPDU-urilor, situație care se înrăutățește la viteze mari. Protocoalele ar trebui proiectate astfel, încât să minimizeze numărul de TPDU-uri, copierile lor repetate și comutările de context. Pentru rețelele gigabit sunt de dorit protocoale simple.

6.8 PROBLEME

1. În exemplele noastre de primitive de transport din fig. 6-2, LISTEN este un apel blocant. Este acest lucru strict necesar? Dacă nu, explicați cum ar putea fi utilizată o primitivă neblocantă. Ce avantaje ar avea aceasta pentru schema descrisă în text?

2. În modelul pe care se bazează fig. 6-4 se presupune că pachetele pot fi pierdute de către nivelul rețea și trebuie deci să fie confirmate individual. Să presupunem că nivelul rețea este 100% fiabil și nu pierde pachete niciodată. Ce modificări sunt necesare (dacă sunt necesare) în fig. 6-4?

3. În ambele părți ale fig. 6-6 este un comentariu conform căruia valoarea SERVER_PORT trebuie să fie aceeași și în client și în server. De ce este acest lucru atât de important?

4. Să presupunem că pentru generarea numerelor de secvență inițiale se utilizează o schemă dirijată de ceas cu un contor de timp de 15 biți. Ceasul generează un impuls la fiecare 100 ms și durata de viață maximă a unui pachet este de 60 sec. Cât de des este necesar să aibă loc o resincronizare
   a) în cel mai rău caz?
   b) atunci când se consumă 240 de numere de secvență pe secundă?

5. De ce este necesar ca timpul maxim de viață al unui pachet, *T*, să fie suficient de mare pentru a acoperi nu numai dispariția pachetului, dar și a confirmării?

6. Să ne imaginăm că pentru stabilirea unei conexiuni se utilizează un protocol cu înțelegere în doi pași și nu unul cu înțelegere în trei pași. Cu alte cuvinte, al treilea mesaj nu mai este necesar. Sunt posibile interblocări în această situație? Dați un exemplu sau arătați că nu există nici o interblocare.

7. Imaginați o problemă generalizată a celor *n* armate, în care acordul dintre oricare două armate este suficient pentru victorie. Există un protocol care îi permite albastrului să câștige?

8. Să considerăm problema recuperării după defectarea unei mașini gazdă (adică fig. 6-18). Dacă intervalul dintre scrierea și trimiterea unei confirmări, sau vice-versa, poate fi făcut relativ scurt, care sunt cele mai bune strategii emițător-receptor pentru minimizarea șansei de defectare a protocolului?

9. Sunt posibile interblocările pentru entitățile transport descrise în text (fig. 6-20)?

10. Din pură curiozitate, programatorul entității transport din fig. 6-20 a decis să pună contoare în interiorul procedurii *sleep*, pentru a colecta astfel statistici despre vectorul *conn*. Între acestea se află și numerele de conexiuni din fiecare dintre cele șapte stări posibile Σn_i (i=1,…, 7). După scrierea unui program FORTRAN serios pentru a analiza datele, programatorul nostru descoperă că relația Σn_i = *MAX_CONN* pare să fie totdeauna adevărată. Există și alți invarianți care să implice doar aceste șapte variabile?

11. Ce se întâmplă dacă utilizatorul entității transport din fig. 6-20 trimite un mesaj de lungime 0? Discutați semnificația răspunsului.

12. Pentru fiecare eveniment care poate apărea în entitatea transport din fig. 6-20, spuneți dacă este sau nu ca utilizatorul să doarmă (eng.: sleep) în starea *transmisie*.

13. Discutați avantajele și dezavantajele creditelor față de protocoalele cu fereastră glisantă.

14. De ce există UDP? Nu ar fi fost suficient ca procesul utilizator să fie lăsat să trimită pachete bloc IP?

15. Se consideră un protocol simplu la nivelul aplicației, construit pe UDP, care permite unui client să recupereze un fișier de la un server la distanță aflat la o adresă bine cunoscută. Clientul trimite mai întâi o cerere cu numele fișierului, iar serverul răspunde cu o secvență de pachete de date conținând diferite părți din fișierul cerut. Pentru a asigura fiabilitate și livrare secvențială, clientul și serverul folosesc un protocol pas-cu-pas. Ignorând problema evidentă a performanței, vedeți vreo problemă la acest protocol? Gândiți-vă atent la posibilitatea terminării bruște a proceselor.

16. Un client trimite cereri de 128 de octeți către un server localizat la 100 km depărtare, printr-un cablu optic de 1 gigabit. Care este eficiența liniei în timpul apelului de procedură la distanță?

17. Să considerăm din nou situația din problema precedentă. Calculați timpul minim posibil de răspuns atât pentru linia de 1 Gbps anterioară cât și pentru o linie de 1 Mbps. Ce concluzie puteți trage?

18. Atât UDP, cât și TCP, folosesc numere de port pentru a identifica entitatea destinație când livrează mesaje. Dați două motive pentru care aceste protocoale au inventat un nou ID abstract (numere de port), în loc să folosească ID-uri de procese, care existau deja când aceste protocoale au fost proiectate.

19. Care este dimensiunea totală a MTU TCP, incluzând supraîncărcarea TCP-ului și IP-ului dar neincluzând supraîncărcarea nivelului legătură de date?

20. Fragmentarea și reasamblarea datagramelor sunt controlate de IP și sunt invizibile TCP-ului. Înseamnă acest lucru că TCP-ul nu trebuie să-și facă griji pentru datele care sosesc în ordine eronată?

21. RTP este utilizat pentru a transmite date audio de calitatea CD-ului, ceea ce înseamnă o pereche de eșantioane pe 16 biți de 44.100 de ori pe secundă, un eșantion pentru fiecare dintre canalele stereo. Câte pachete pe secundă trebuie să transmită RTP?

22. Ar fi posibil să fie plasat cod RTP în nucleul sistemului de operare, alături de cod UDP? Explicați răspunsul.

23. Un proces de pe mașina 1 a fost asociat portului *p* și un proces de pe mașina 2 a fost asociat portului *q*. Este posibil ca între cele două porturi să fie deschise mai multe conexiuni TCP în același timp?

24. În fig. 6-29 am văzut că alături de câmpul *Acknowledgement* de 32 de biți, este un bit *ACK* în al patrulea cuvânt. Acesta adaugă cu adevărat ceva? De ce da, sau de ce nu?

25. Informația utilă maximă dintr-un segment TCP este de 65495 octeți. De ce a fost ales un număr atât de straniu?

26. Descrieți două moduri de a ajunge în starea *SYN RCVD* din fig. 6-28.

27. Prezentați un dezavantaj potențial al algoritmului Nagle atunci când este utilizat într-o rețea puternic congestionată.

28. Să considerăm efectul utilizării startului lent pe o linie cu timpul circuitului dus-întors de 10 ms și fără congestie. Fereastra receptorului este de 24 Kocteți și dimensiunea maximă a segmentului este de 2 Kocteți. Cât timp trebuie să treacă înainte ca prima fereastră completă să poată fi trimisă?

29. Să presupunem că fereastra de congestie TCP este de 18 Kocteți și apare o depășire de timp. Cât de mare va fi fereastra dacă următoarele patru rafale de transmisie reușesc? Se presupune că dimensiunea maximă a segmentului este de 1 Koctet.

30. Dacă timpul circuitului TCP dus-întors, RTT, este la un moment dat 30 ms și următoarele confirmări sosesc după 26, 32 și, respectiv, 24 ms, care este noul RTT estimat folosind algoritmul Jacobson? Utilizați α=0.9.

31. O mașină TCP trimite cadre de 65535 octeți pe un canal de 1 Gbps pentru care întârzierea pe un singur sens este de 10 ms. Care este productivitatea maximă care poate fi atinsă? Care este eficiența liniei?

32. Care este cea mai mare viteză a liniei la care o mașină gazdă poate transmite pachete TCP de 1500 octeți cu un timp de viață care poate fi de maxim 120 sec fără ca numărul de secvență să se repete? Luați în considerare supraîncărcarea de la TCP, IP și Ethernet. Presupuneți că se pot transmite continuu cadrele Ethernet.

33. Într-o rețea care are dimensiunea maximă a TPDU-urilor de 128 octeți, timpul maxim de viață al unui TPDU de 30 sec și numărul de secvență de 8 biți, care este rata maximă de date per conexiune?

34. Să presupunem că măsurați timpul necesar recepționării unui TPDU. Atunci când apare o întrerupere, se citește timpul sistem în milisecunde. Când TPDU-ul este complet prelucrat, se citește din nou timpul sistem. S-au înregistrat 0 ms de 270000 de ori și 1 de 730000 de ori. Care este timpul de recepție al unui TPDU?

35. Un procesor execută instrucțiuni la o viteză de 1000 MIPS. Informația poate fi copiată câte 64 de biți odată, fiecare copiere a unui cuvânt costând zece instrucțiuni. Dacă un pachet recepționat trebuie să fie copiat de patru ori, poate sistemul să facă față unei linii de 1 Gbps? Pentru a simplifica, presupunem că toate instrucțiunile, inclusiv acelea de citire/scriere din memorie, rulează la viteza maximă de 1000 MIPS.

36. Pentru a evita problema revenirii numerelor de secvență la valori inițiale în timp ce există încă pachete vechi, s-ar putea utiliza numere de secvență pe 64 de biți. Cu toate acestea, teoretic, un cablu optic poate opera la 75 Tbps. Care este durata maximă de viață pe care trebuie să o aibă un pachet pentru ca viitoarele rețele de 75 Tbps să nu se lovească de aceeași problemă a revenirii numerelor de secvență chiar și în cazul reprezentării lor pe 64 biți? Presupuneți, ca și TCP-ul, că fiecare octet are propriul său număr de secvență.

37. Dați un avantaj al RPC pe UDP față de TCP tranzacțional. Dați un avantaj al T/TCP față de RPC.

38. În fig. 6-40(a), vedem că sunt necesare 9 pachete pentru a realiza RPC-ul. Există situații în care sunt necesare exact 10 pachete?

39. În secțiunea 6.6.5 am calculat că o linie gigabit livrează unei mașini gazdă 80000 de pachete pe secundă, permițându-i doar 6250 de instrucțiuni pentru a prelucra un pachet și lăsând doar jumătate din capacitatea procesorului pentru aplicații. Acest calcul presupune un pachet de 1500 octeți. Refaceți calculul pentru pachetele ARPANET de dimensiune de 128 octeți. În ambele cazuri, presupuneți că dimensiunile pachetelor date includ toate supraîncărcările.

40. Pentru o rețea care operează la 1 Gbps pe o distanță de 4000 km, factorul limitator nu este dat de lărgimea de bandă, ci de întârziere. Considerăm un MAN cu sursa și destinația situate în medie la 20 km una de cealaltă. La ce viteză de date întârzierea circuitului dus-întors datorată vitezei luminii egalează întârzierea de transmisie pentru un pachet de 1 Koctet?

41. Calculați produsul dintre întârziere și lățimea de bandă pentru următoarele rețele: (1) T1 (1,5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps) și (4) STS-3 (155 Mbps). Presupuneți RTT de 100 ms. Amintiți-vă ca antetul TCP-lui are 16 biți rezervați pentru Dimensiunea Ferestrei. Care sunt implicațiile din punctul de vedere al calculelor voastre?

42. Care este produsul dintre întârziere și lățimea de bandă pentru un canal de 50 Mbps pe un satelit geostaționar? Dacă pachetele sunt toate de 1500 de octeți (incluzând supraîncărcarea), cât de mare ar trebui să fie fereastra în pachete?

43. Serverul de fișiere din fig. 6-6 este departe de a fi perfect și i-ar fi folositoare câteva îmbunătățiri. Faceți următoarele modificări:
a) Dați clientului un al treilea argument care specifică un interval de octeți.
b) Adăugați un flag –w de client care permite fișierului să fie scris pe server.

44. Modificați programul din fig. 6-20 pentru a asigura revenirea din erori. Adăugați un nou tip de pachet, *reset*, care poate ajunge doar după ce conexiunea a fost deschisă de ambele părți și n-a fost închisă de niciuna. Acest eveniment, care are loc simultan la ambele capete ale conexiunii, indică faptul că orice pachet care era în tranzit a fost sau distrus, sau livrat, în orice caz el nemaiaflându-se în subrețea.

45. Scrieți un program care simulează controlul tampoanelor într-o entitate transport, utilizând un flux de control cu fereastră glisantă și nu un control al fluxului cu credite, ca în fig. 6-20. Lăsați procesele de pe nivelul superior să deschidă conexiuni, să trimită date și să închidă conexiuni în mod aleatoriu. Pentru a păstra programul cât mai simplu, faceți ca toată informația să călătorească doar de la mașina A la mașina B și deloc în sens invers. Experimentați cu diferite strategii de alocare a tampoanelor la nivelul mașinii B, ca, de exemplu, tampoane dedicate unei anume conexiuni față de tampoane preluate dintr-un depozit comun și măsurați productivitatea totală atinsă în ambele cazuri.

46. Proiectați și implementați un sistem de discuții care permite mai multor grupuri de utilizatori să discute. Coordonatorul discuțiilor se află la o adresă de rețea bine cunoscută, folosește UDP pentru comunicarea cu clienții de discuții, setează serverele de discuții pentru fiecare sesiune de discuții, și menține un director de sesiuni de discuții. Este un server de discuții pentru fiecare sesiune de discuții. Un server de discuții folosește TCP pentru comunicație cu clienți. Un client de discuții permite utilizatorilor să pornească, să intre sau să iasă dintr-o sesiune de discuții. Proiectați și implementați codul pentru coordonator, server și client.

7

NIVELUL APLICAȚIE

După ce am epuizat toate preliminariile putem aborda nivelul unde pot fi găsite toate aplicațiile. Nivelurile de sub nivelul aplicație servesc la asigurarea unui transport sigur, dar nu îndeplinesc nici o funcție concretă pentru utilizatori. În acest capitol vom studia câteva aplicații reale.

Totuși, chiar și la nivelul aplicație, apare necesitatea unor protocoale-suport care să permită funcționarea aplicațiilor reale. Înainte de a începe studiul aplicațiilor, ne vom ocupa de unul dintre acestea. Subiectul în discuție este DNS, care se ocupă de convențiile de nume în Internet. După aceea vom examina trei aplicații reale: poșta electronică, World Wide Web (rom.: rețea de întindere planetară) și, în final, multimedia.

7.1 DNS - SISTEMUL NUMELOR DE DOMENII

Cu toate că teoretic programele ar putea să se refere la sistemele gazdă, la cutiile poștale și la alte resurse prin adresa lor de rețea (de exemplu prin adresa IP), aceste adrese sunt greu de memorat de către oameni. De asemenea, în trimiterea de poștă electronică la *tana@128.111.24.41* ar însemna că dacă furnizorul de servicii Internet sau organizația Tanei mută serverul de poștă pe o mașină diferită, cu o adresă IP diferită, adresa ei de e-mail se va schimba. De aceea au fost introduse nume ASCII pentru a separa numele mașinilor de adresele mașinilor. În acest fel, adresa Tanei ar putea fi ceva de genul *tana@art.ucsb.edu*. Cu toate acestea, rețeaua înțelege numai adrese numerice, deci este necesar un mecanism care să convertească șirurile ASCII în adrese de rețea. În secțiunile următoare se va studia cum este realizată această conversie în Internet.

Încă de la ARPANET exista un fișier *host.txt* care cuprindea toate sistemele gazdă și adresele lor IP. În fiecare noapte, toate gazdele îl preluau de la situl unde era păstrat. Pentru o rețea formată din câteva sute de mașini mari, cu divizarea timpului, această abordare era destul de rezonabilă.

Totuși, atunci când la rețea au fost conectate mii de stații de lucru, toți și-au dat seama că această abordare nu putea să funcționeze la nesfârșit. În primul rând dimensiunea fișierului ar deveni prea mare. Cu toate acestea și chiar mai important, conflictele de nume de sisteme gazdă ar apărea în permanență dacă nu ar fi administrate centralizat, ceva de negândit într-o rețea internațională de dimensiuni uriașe din cauza încărcării și a latenței. Pentru a rezolva aceste probleme, a fost inventat **DNS** (**Domain Name System** - Sistemul numelor de domenii).

Esența DNS-ului constă într-o schemă ierarhică de nume de domenii și a unui sistem de baze de date distribuite pentru implementarea acestei scheme de nume. În principal este utilizat pentru a pune în corespondență numele sistemelor gazdă și adresele destinațiilor de e-mail cu adresele IP, dar poate fi utilizat și pentru alte scopuri. DNS este definit în RFC-urile 1034 și 1035.

Foarte pe scurt, DNS este utilizat după cum urmează. Pentru a stabili corespondența dintre un nume și o adresă IP, programul de aplicație apelează o procedură de bibliotecă numită **resolver**, transferându-i numele ca parametru. Putem vedea un exemplu de resolver, *gethostbyname*, în fig. 6-6. Resolver-ul trimite un pachet UDP la serverul DNS local, care caută numele și returnează adresa IP către resolver, care o returnează apelantului. Înarmat cu adresa IP, programul poate stabili o conexiune TCP cu destinația sau îi poate trimite pachete UDP.

7.1.1 Spațiul de nume DNS

Administrarea unui volum mare de nume în permanentă schimbare nu este o problemă prea ușoară. În sistemul poștal, administrarea numelor este realizată impunând ca pe o scrisoare să se specifice (implicit sau explicit) țara, statul sau provincia, orașul, strada și restul adresei destinatarului. Utilizând o astfel de adresare ierarhică, nu există nici o confuzie între Marvin Anderson de pe Main St. din White Plains, N.Y. și Marvin Anderson de pe Main St. din Austin, Texas. DNS lucrează în același mod.

Conceptual, Internetul este divizat în peste 200 **domenii** de nivel superior, fiecare domeniu cuprinzând mai multe sisteme gazdă. Fiecare domeniu este partiționat în subdomenii și acestea sunt, la rândul lor, partiționate ș.a.m.d. Toate aceste domenii pot fi reprezentate ca un arbore, așa cum se arată în fig. 7-1. Frunzele arborelui reprezintă domenii care nu au subdomenii (dar, bineînțeles, conțin sisteme). Un domeniu frunză poate conține un singur sistem gazdă sau poate reprezenta o firmă, deci să conțină mii de sisteme gazdă.

Domeniile de pe primul nivel se împart în două categorii: generice și de țări. Domeniile generice sunt *com* (comercial), *edu* (instituții educaționale), *gov* (guvernul federal al SUA), *int* (organizații internaționale), *mil* (forțele armate ale SUA), *net* (furnizori Internet) și *org* (organizații nonprofit). Domeniile de țări includ o intrare pentru fiecare țară, cum se definește în ISO 3166.

În noiembrie 2000, ICANN a aprobat patru domenii de nivel superior noi, de interes general, și anume, *biz* (afaceri), *info* (informații), *name* (nume de persoane), și *pro* (profesii, ca de exemplu doctori sau avocați). În plus, au fost introduse trei domenii de nivel superior cu caracter specializat, cerute de către anumite industrii. Acestea sunt *aero* (industria aerospațială), *coop* (cooperative), și *museum* (muzee). În viitor vor fi adăugate alte domenii superioare.

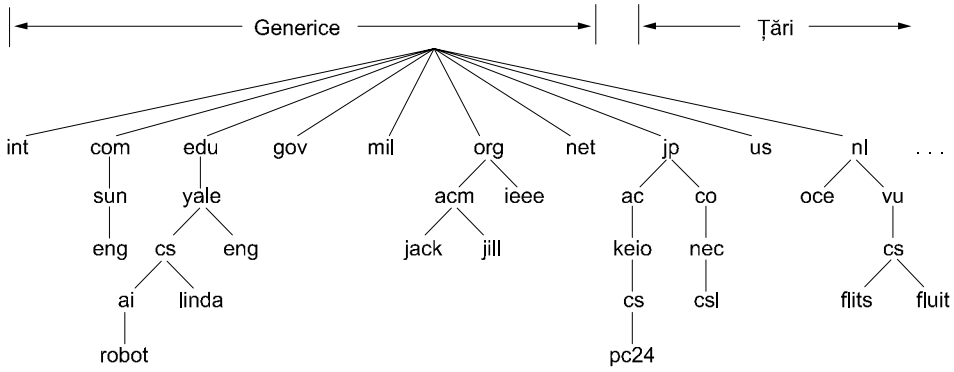


Fig. 7-1. O porțiune a spațiului numelor de domenii din Internet.

Pe de altă parte, pe măsură ce Internetul devine mai comercial, el devine și mai discutabil. Să luăm domeniul *pro*, de exemplu, care a fost proiectat pentru profesioniștii atestați. Dar cine este un profesionist? Și de cine este atestat? Dar cum rămâne cu fotografii, profesorii de pian, magicienii, instalatorii, frizerii, exterminatorii, artiștii de tatuaje, mercenarii și prostituatele? Sunt acestea meserii și sunt acceptabile pentru domeniile *pro*? Și daca da, cine atestă diverși practicieni?

În general, obținerea unui domeniu de nivel secundar, ca de exemplu *nume-al-companiei.com*, este ușoară. Pur și simplu este necesară doar consultarea serviciului de înregistrare al nivelului superior corespunzător (*com* în acest caz) pentru a vedea dacă numele dorit este disponibil și nu aparține altcuiva. Dacă nu sunt probleme, solicitantul plătește o mică taxă anuală și primește numele. Până acum, cam toate cuvintele comune (din engleză) au fost luate în domeniul *com*. Încercați nume de articole casnice, animale, plante, părți ale corpului etc. Aproape toate sunt luate.

Fiecare domeniu este identificat prin calea în arbore de la el la domeniul (fără nume) rădăcină. Componentele sunt separate prin puncte (pronunțat „dot"). Astfel, departamentul tehnic de la Sun Microsystems ar putea fi *eng.sun.com*, în loc de numele în stil UNIX */com/sun/eng*. De notat că această numire ierarhică face ca *eng.sun.com* să nu intre în conflict cu posibila utilizare a lui *eng* din *eng.yale.edu*, care ar putea fi folosit pentru departamentul de limba engleză de la Yale.

Numele de domenii pot fi absolute sau relative. Un nume absolut de domeniu se termină cu un punct (de exemplu, *eng.sun.com.*), în timp ce unul relativ nu. Numele relative trebuie interpretate în context pentru a le determina înțelesul adevărat. În ambele cazuri, un nume de domeniu se referă la un anumit nod din arbore și la toate nodurile de sub el.

Numele de domenii nu fac distincție între litere mici și litere mari, astfel *edu*, *Edu*, sau *EDU* înseamnă același lucru. Componentele numelor pot avea o lungime de cel mult 63 caractere, iar întreaga cale de nume nu trebuie să depășească 255 de caractere.

În principiu, domeniile pot fi inserate în arbore în două moduri diferite. De exemplu, *cs.yale.edu* ar putea la fel de bine să fie inclus în domeniul țării *us* ca *cs.yale.ct.us*. În practică, totuși, aproape toate organizațiile din Statele Unite sunt repartizate după criteriul generic, iar aproape toate din afara Statelor Unite fac parte din domeniul țării lor. Nu există nici o regulă împotriva înregistrării sub două domenii de nivel superior, însă puține organizații în afară de cele multinaționale o fac (de exemplu, *sony.com* și *sony.nl*).

Fiecare domeniu controlează cum sunt alocate domeniile de sub el. De exemplu, Japonia are domeniile *ac.jp* și *co.jp* echivalente cu *edu* și *com*. Olanda nu face nici o distincție și pune toate orga-

nizațiile direct sub *nl*. Astfel următoarele trei nume sunt toate departamente de calculatoare (computer science) din universități:

1. *cs.yale.edu* (Universitatea Yale din Statele Unite).
2. *cs.vn.nl* (Vrije Universiteit în Olanda).
3. *cs.keio.ac.jp* (Universitatea Keio din Japonia).

Pentru a crea un nou domeniu, se cere permisiunea domeniului în care va fi inclus. De exemplu, dacă un grup VLSI de la Yale dorește să fie cunoscut ca *vlsi.cs.yale.edu,* acesta are nevoie de permisiunea celui care administrează *cs.yale.edu*. Similar, dacă este acreditată o nouă universitate, să zicem Universitatea din Northern South Dakota, ea trebuie să ceară administratorului domeniului *edu* să-i atribuie *unsd.edu*. În acest mod sunt evitate conflictele de nume și fiecare domeniu poate ține evidența tuturor subdomeniilor sale. Odată ce un nou domeniu a fost creat și înregistrat, el poate crea subdomenii, cum ar fi *cs.unsd.edu*, fără a cere permisiunea de la cineva din partea superioară a arborelui.

Atribuirea de nume respectă granițele organizaționale, nu pe cele ale rețelelor fizice. De exemplu, dacă departamentele de știința calculatoarelor și de inginerie electrică sunt localizate în aceeași clădire și folosesc aceeași rețea locală, ele pot avea totuși domenii distincte. Similar, dacă departamentul de știința calculatoarelor este împărțit în două clădiri (Babbage Hall și Turing Hall), toate sistemele gazdă din ambele clădiri aparțin aceluiași domeniu.

7.1.2 Înregistrări de resurse

Fiecărui domeniu, fie că este un singur calculator gazdă, fie un domeniu de nivel superior, îi poate fi asociată o mulțime de înregistrări de resurse (**resource records**). Pentru un singur sistem gazdă, cea mai obișnuită înregistrare de resursă este chiar adresa IP, dar există multe alte tipuri de înregistrări de resurse. Atunci când un resolver trimite un nume de domeniu către un DNS, ceea ce va primi ca răspuns sunt înregistrările de resurse asociate acelui nume. Astfel, adevărata funcție a DNS este să realizeze corespondența dintre numele de domenii și înregistrările de resurse.

O înregistrare de resursă este un 5-tuplu. Cu toate că, din rațiuni de eficiență, înregistrările de resurse sunt codificate binar, în majoritatea expunerilor ele sunt prezentate ca text ASCII, câte o înregistrare de resursă pe linie. Formatul pe care îl vom utiliza este următorul:

```
Nume_domeniu Timp_de_viață Clasă Tip Valoare
```

Nume_domeniu (domain_name) precizează domeniul căruia i se aplică această înregistrare. În mod normal există mai multe înregistrări pentru fiecare domeniu și fiecare copie a bazei de date păstrează informații despre mai multe domenii. Acest câmp este utilizat ca cheie de căutare primară pentru a satisface cererile. Ordinea înregistrărilor în baza de date nu este semnificativă.

Câmpul *Timp_de_viață* (time_to_live) dă o indicație despre cât de stabilă este înregistrarea. Informația care este foarte stabilă are asigurată o valoare mare, cum ar fi 86400 (numărul de secunde dintr-o zi). Informației instabile îi este atribuită o valoare mică, cum ar fi 60 (1 minut). Vom reveni la acest punct mai târziu, când vom discuta despre utilizarea memoriei ascunse.

Al treilea câmp dintr-o înregistrare de resursă este *Clasa* (class). Pentru informațiile legate de Internet este tot timpul *IN*. Pentru alte informații pot fi folosite alte coduri, însă în practică acestea se întâlnesc rar.

Câmpul *Tip* (type) precizează tipul înregistrării. Cele mai importante tipuri sunt prezentate în fig. 7-2.

Tip	Semnificație	Valoare
SOA	Start autoritate	Parametrii pentru această zonă
A	Adresa IP a unui sistem gazdă	Întreg pe 32 de biți
MX	Schimb de poștă	Prioritate, domeniu dispus să accepte poștă electronică
NS	Server de Nume	Numele serverului pentru acest domeniu
CNAME	Nume canonic	Numele domeniului
PTR	Pointer	Pseudonim pentru adresa IP
HINFO	Descriere sistem gazdă	Unitate centrală și sistem de operare în ASCII
TXT	Text	Text ASCII neinterpretat

Fig. 7-2. Principalele tipuri de înregistrări de resurse DNS.

O înregistrare *SOA* furnizează numele sursei primare de informații despre zona serverului de nume (descrisă mai jos), adresa de e-mail a administratorului, un identificator unic și diverși indicatori și contoare de timp.

Cel mai important tip de înregistrare este înregistrarea *A* (adresă). Ea păstrează adresa IP de 32 de biți a unui sistem gazdă. Fiecare sistem gazdă Internet trebuie să aibă cel puțin o adresă IP, astfel încât alte mașini să poată comunica cu el. Unele sisteme gazdă au două sau mai multe conexiuni în rețea, caz în care vor avea câte o înregistrare de tip *A* pentru fiecare conexiune (și astfel pentru fiecare adresă IP).

Următoarea ca importanță este înregistrarea *MX*. Aceasta precizează numele sistemului gazdă pregătit să accepte poșta electronică pentru domeniul specificat. El este folosit deoarece nu toate mașinile sunt pregătite să accepte poșta electronică pentru domeniul specificat. Dacă cineva vrea să-i trimită un e-mail, de exemplu, lui *bill@microsoft.com*, sistemul care trimite trebuie să găsească un server la *microsoft.com* dispus să accepte e-mail. Înregistrarea *MX* poate să furnizeze această informație.

Înregistrările *NS* specifică serverele de nume. De exemplu, fiecare bază de date DNS are în mod normal o înregistrare *NS* pentru fiecare domeniu de pe primul nivel, astfel încât, de exemplu, poșta electronică să poată fi trimisă în zone îndepărtate ale arborelui de nume. Vom reveni la acest aspect mai târziu.

Înregistrările *CNAME* permit crearea pseudonimelor. De exemplu, o persoană familiarizată cu atribuirea numelor în Internet, care dorește să trimită un mesaj unei persoane al cărei nume de conectare la un sistem de calcul din departamentul de calculatoare de la M.I.T. este *paul* poate presupune că adresa *paul@cs.mit.edu* este corectă. De fapt această adresă nu este corectă, deoarece domeniul departamentului de calculatoare de la M.I.T. este *lcs.mit.edu*. Totuși, ca un serviciu pentru cei care nu știu acest lucru, M.I.T. poate crea o intrare *CNAME,* pentru a dirija persoanele și programele în direcția corectă. O astfel de intrare poate fi:

```
cs.mit.edu 86400 IN CNAME lcs.mit.edu
```

Ca și *CNAME*, *PTR* se referă la un alt nume. Totuși, spre deosebire de *CNAME,* care este în realitate numai o macro-definiție, *PTR* este un tip de date DNS a cărui interpretare depinde de contextul în care este utilizat. În practică este aproape întotdeauna utilizat pentru asocierea unui nume cu o adresă IP, pentru a permite căutarea adresei IP și obținerea numelui mașinii corespunzătoare. Acestea se numesc **căutări inverse** (**reverse lookups**).

Înregistrările *HINFO* permit aflarea tipului de mașină și de sistem de operare cărora le corespunde domeniul. În sfârșit, înregistrările *TXT* permit domeniilor să se autoidentifice într-un mod arbitrar. Aceste două tipuri de înregistrări sunt introduse pentru ușurința utilizatorului. Nici una

dintre ele nu este necesară, astfel încât programele nu pot conta pe obținerea lor (și probabil că dacă le obțin nu le pot trata).

În final ajungem la câmpul *Valoare*. Acest câmp poate fi un număr, un nume de domeniu sau un șir ASCII. Semantica depinde de tipul de înregistrare. O scurtă descriere a câmpurilor *Valoare* pentru fiecare dintre principalele tipuri de înregistrări este dată în fig. 7-2.

Un exemplu de informație ce se poate găsi în baza de date DNS a unui domeniu este prezentat în fig. 7-3. Această figură prezintă o parte (semi-ipotetică) a bazei de date pentru domeniul *cs.vu.nl* prezentat în fig. 7-1. Baza de date conține șapte tipuri de înregistrări de resurse.

```
;Baza de date pentru cs.vu.nl
cs.vu.nl.        86400   IN  SOA      star boss (9527, 7200, 7200, 241920, 86400)
cs.vu.nl.        86400   IN  TXT      „Divisie Wiskunde en Informatica."
cs.vu.nl.        86400   IN  TXT      „Vrije Universiteit Amsterdam."
cs.vu.nl.        86400   IN  MX       1 zephyr.cs.vu.nl.
cs.vu.nl.        86400   IN  MX       2 top.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  HINFO    Sun Unix
flits.cs.vu.nl.  86400   IN  A        130.37.16.112
flits.cs.vu.nl.  86400   IN  A        192.31.231.165
flits.cs.vu.nl.  86400   IN  MX       1 flits.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX       2 zephyr.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX       3 top.cs.vu.nl.
www.cs.vu.nl.    86400   IN  CNAME    star.cs.vu.nl.
ftp.cs.vu.nl.    86400   IN  CNAME    zephyr.cs.vu.nl.
rowboat                  IN  A        130.37.56.201
                         IN  MX       1 rowboat
                         IN  MX       2 zephyr
                         IN  HINFO    Sun Unix

little-sister            IN  A        130.37.62.23
                         IN  HINFO    Mac MacOS

laserjet                 IN  A        192.31.231.216
                         IN  HINFO    „HP Laserjet IIISi" Proprietary
```

Fig. 7-3. O parte dintr-o posibilă bază de date DNS pentru *cs.vu.nl*.

Prima linie necomentată din fig. 7-3 dă câteva informații de bază despre domeniu, de care nu ne vom ocupa. Următoarele două linii furnizează informații textuale despre amplasarea domeniului. Urmează două intrări care specifică primul și al doilea loc unde se încearcă să se livreze poșta electronică trimisă pentru *persoana@cs.vu.nl*. Mai întâi se încearcă trimiterea la mașina *zephyr.* Dacă aceasta eșuează, atunci trebuie să se încerce la *top*.

După o linie liberă, adăugată numai pentru claritate, urmează linii care spun că *flits* este o stație de lucru Sun care lucrează sub UNIX și se specifică ambele sale adrese IP. Urmează trei variante de tratare a poștei electronice trimise la *flits.cs.vu.nl.* Prima alegere este, în mod natural, chiar *flits*, iar dacă nu se reușește, se încearcă la *zephyr* și apoi la *top*. Urmează un pseudonim, *www.cs.vu.nl,* astfel ca această adresă să poată fi utilizată fără a specifica o anumită mașină. Crearea acestui pseudonim permite ca *cs.vu.nl* să schimbe serverul www fără invalidarea adresei folosite în mod curent pentru adresarea lui. Un argument similar este valabil pentru *ftp.cs.vu.nl.*

Următoarele patru linii conțin o înregistrare tipică pentru o stație de lucru, în acest caz *rowboat. cs.vu.nl*. Informația furnizează adresa IP, destinația primară și secundară pentru poșta electronică și

informații despre mașină. Urmează o intrare pentru un sistem non-UNIX care nu este capabil să primească poșta el însuși, urmat de o intrare pentru o imprimantă laser conectată la Internet.

Ceea ce nu este arătat (și nu există în acest fișier) sunt adresele IP utilizate pentru a căuta adresele domeniilor de pe primul nivel. Acestea sunt necesare pentru a căuta sistemele gazdă aflate la distanță, dar, deoarece ele nu fac parte din domeniul *cs.vu.nl*, nu se găsesc în acest fișier. Ele sunt furnizate de serverele rădăcină ale căror adrese IP sunt prezentate în fișierul de configurare a sistemului și sunt încărcate în memoria ascunsă DNS atunci când este pornit serverul DNS. Există cam o duzină de servere rădăcină în lume și fiecare știe adresele IP ale tuturor celorlalte servere de domenii de nivel superior. Astfel, dacă o mașină știe adresa IP a cel puțin unuia din serverele rădăcină, el poate căuta orice nume DNS.

7.1.3 Servere de nume

Teoretic, un singur server de nume poate conține întreaga bază de date DNS și poate să răspundă tuturor cererilor. În practică, acest server poate fi atât de încărcat, încât să devină de neutilizat. În afară de aceasta, dacă se defectează, va fi afectat întregul Internet.

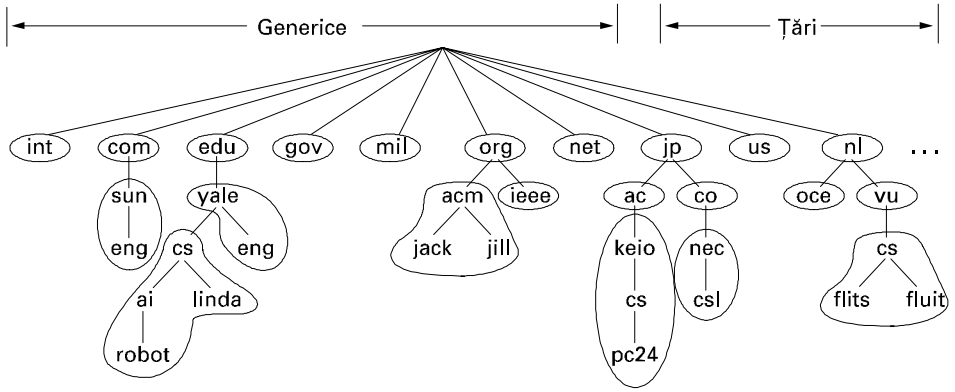


Fig. 7-4. O parte din spațiul numelor DNS prezentând împărțirea în zone.

Pentru a evita problemele asociate cu existența unei singure surse de informație, spațiul de nume DNS este împărțit în **zone** care nu se suprapun. O posibilă cale de împărțire a spațiului de nume din fig. 7-1 este arătată în fig. 7-4. Fiecare zonă conține câte o parte a arborelui precum și numele servere-lor care păstrează informația autorizată despre acea zonă. În mod normal, o zonă va avea un server de nume primar, care preia informația dintr-un fișier de pe discul propriu și unul sau mai multe servere de nume secundare, care iau informațiile de pe serverul primar. Pentru a îmbunătăți fiabilitatea, unele servere pentru o zonă pot fi plasate în afara zonei.

Plasarea limitelor unei zone este la latitudinea administratorului ei. Această decizie este luată în mare parte bazându-se pe câte servere de nume sunt dorite și unde să fie plasate. De exemplu, în fig. 7-4, Yale are un server pentru *yale.edu* care administrează *eng.yale.edu,* dar nu și *cs.yale.edu,* care este o zonă separată cu propriile servere de nume. O astfel de decizie poate fi luată atunci când un departament ca cel de engleză nu dorește să aibă propriul server de nume, în schimb departamentul de calculatoare dorește. În consecință *cs.yale.edu* este o zonă separată, în timp ce zona *eng. yale.edu* nu este separată.

Atunci când un resolver are o cerere referitoare la un nume de domeniu, el transferă cererea unuia din serverele locale de nume. Dacă domeniul căutat este sub jurisdicția serverului de nume, cum ar fi *ai.cs.yale.edu,* care este sub *cs.yale.edu*, el reîntoarce înregistrări de resurse autorizate. O **înregistrare autorizată** (**authoritative record**) este cea care vine de la autoritatea care administrează înregistrarea și astfel este întotdeauna corectă. Înregistrările autorizate se deosebesc de înregistrările din memoria ascunsă, care pot fi expirate.

Dacă, totuși, domeniul se află la distanță, iar local nu este disponibilă nici o informație despre domeniul cerut, atunci serverul de nume trimite un mesaj de cerere la serverul de nume de pe primul nivel al domeniului solicitat. Pentru a clarifica acest proces să considerăm exemplul din fig. 7-5. Aici resolverul de pe *flits.cs.vu.nl* dorește să știe adresa IP a sistemului gazdă *linda.cs.yale.edu*. În pasul 1 trimite o cerere la serverul de nume local *cs.vu.nl*. Această cerere conține numele de domeniu căutat, tipul (*A*) și clasa (*IN*).

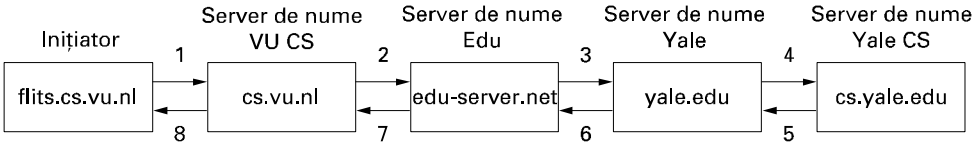


Fig. 7-5. Modul în care un resolver caută un nume aflat la distanță, în opt pași.

Să presupunem că serverul local de nume nu a avut niciodată o cerere pentru acest domeniu și nu știe nimic despre el. Poate solicita informații de la câteva servere de nume din apropriere, dar dacă nici unul dintre ele nu știe, va trimite un pachet UDP la serverul pentru *edu* specificat în baza de date (vezi fig. 7-5), *edu-server.net*. Este puțin probabil ca acest server să cunoască adresa *linda.cs.yale.edu* și probabil nu cunoaște nici adresa *cs.yale.edu,* dar trebuie să cunoască adresele fiilor din subarbore, astfel că va transmite cererea la serverul de nume *yale.edu* (pas 3). Acesta va transmite cererea mai departe către *cs.yale.edu* (pas 4), care trebuie să aibă înregistrările autorizate de resurse. Deoarece fiecare cerere este de la un client la un server, înregistrarea de resursă cerută parcurge pașii 5 până la 8.

Odată ce aceste înregistrări de resurse ajung înapoi la serverul de nume *cs.vu.nl*, ele vor fi depuse în memoria ascunsă, pentru a fi folosite ulterior. Totuși, această informație nu este autorizată, deoarece orice schimbare făcută la *cs.yale.edu* nu se va propaga spre toate serverele care au folosit-o. Din acest motiv intrările în memoria ascunsă nu ar trebui să aibă viață prea lungă. Acesta este motivul pentru care câmpul *Timp_de_viață* este inclus în fiecare înregistrare de resursă. El informează serverele de nume aflate la distanță cât timp să mențină înregistrările în memoria ascunsă. Dacă o anumită mașină are de ani de zile aceeași adresă IP, această informație ar putea fi păstrată timp de o zi. Pentru informații mai volatile este mai sigur ca înregistrările să fie eliminate după câteva secunde sau un minut.

De menționat că metoda de interogare descrisă aici este cunoscută ca metoda de **interogare recursivă** (**recursive query**), deoarece fiecare server care nu are informația cerută o caută în altă parte și raportează. Este posibilă și o altă variantă. În acest caz, atunci când o cerere nu poate fi rezolvată local, cererea eșuează, dar este întors numele următorului server de pe calea ce trebuie încercată. Unele servere nu implementează interogarea recursivă și întorc întotdeauna numele următorului server la care să se încerce.

De asemenea merită menționat faptul că atunci când un client DNS nu reușește să primească un răspuns înainte de expirarea timpului de căutare, data viitoare va încerca un alt server. Se presupune că serverul este probabil nefuncțional, nu că cererea sau răspunsul s-au pierdut.

Deși DNS este foarte important pentru funcționarea corectă a Internetului, el nu face decât să pună în corespondență nume simbolice de mașini cu adresele lor IP. El nu ajută la localizarea oamenilor, resurselor, serviciilor sau obiectelor în general. Pentru localizarea acestora a fost definit un alt serviciu director, numit **LDAP** (**Lightweight Directory Access Protocol**, rom.: **Protocol de acces la cataloage simplificate**). El este o versiune simplificată a serviciului de cataloage OSI X.500, descris în RFC 2251. El organizează informația sub formă de arbore și permite căutări pe diferite componente. Poate fi privit ca o carte de telefon obișnuită (de tipul „pagini albe"). Nu o să intrăm în amănunte referitoare la el în această carte, însă puteți găsi mai multe informații în (Weltman și Dahbura, 2000).

7.2 POȘTA ELECTRONICĂ

Poșta electronică, sau **e-mail**, cum este cunoscută de către numeroșii săi admiratori, există de peste două decenii. Înainte de 1990, era folosită în special în mediul academic. În timpul anilor 1990, a devenit cunoscută publicului larg și a crescut exponențial până la punctul în care numărul de mesaje electronice trimise pe zi este acum mult mai mare decât numărul de scrisori tradiționale (adică pe hârtie).

E-mail-ul, ca majoritatea celorlalte forme de comunicare, are convențiile și stilurile sale proprii. În particular, el este foarte neprotocolar și are un prag de folosire foarte scăzut. Oamenii care n-ar visa niciodată să sune la telefon sau chiar să scrie o scrisoare unei Persoane Foarte Importante nu ezită o secundă să trimită un e-mail neglijent.

E-mail-ul este plin de jargoane precum BTW (By The Way - apropo), ROTFL (Rolling On The Floor Laughing – a se tăvăli pe jos de râs) și IMHO (In My Humble Opinion - după umila mea părere). Mulți oameni folosesc în e-mail-urile lor câteva caractere ASCII numite **smileys** sau **emoticons** (fața zâmbitoare și fața tristă). Câteva din cele mai interesante sunt reproduse în fig. 7-6. Pentru cei mai mulți, rotirea cărții cu 90 de grade în sensul acelor de ceasornic, le va face mai clare. Pentru o cărticică cu peste 650 smileys, vedeți (Sanderson și Dougherty, 1993).

Smiley	Semnificație	Smiley	Semnificație	Smiley	Semnificație
:-)	Sunt fericit	=|:-)	Abe Lincoln	:+)	Nas mare
:-(	Sunt trist, supărat	=):-)	Unchiul Sam	:-))	Gușă
:-|	Sunt apatic	*<:-)	Moș Crăciun	:-{)	Mustață
;-)	Fac cu ochiul	<:-(	Dunce	#:-)	Păr încurcat
:-(0)	Țip	(-:	Australian	8-)	Poartă ochelari
:-(*)	Vomit	:-)X	Om cu papion	C:-)	Cap mare

Fig. 7-6. Câteva smileys. Nu vor fi în examenul final :-)

Primele sisteme de poștă electronică constau pur și simplu din protocoale de transfer de fișiere, cu convenția ca prima linie a fiecărui mesaj (adică fișier) să conțină adresa receptorului. Cu timpul, limitările acestei abordări au devenit din ce în ce mai evidente. O parte dintre neajunsuri erau:

1. Trimiterea unui mesaj către un grup de persoane era incomodă. Managerii au nevoie adesea de această facilitate pentru a trimite note și rapoarte tuturor subordonaților.
2. Mesajele nu aveau structură internă, făcând astfel dificilă prelucrarea lor cu ajutorul calculatorului. De exemplu, dacă un mesaj trimis mai departe era inclus în corpul altui mesaj, extragerea părții incluse din mesajul primit era dificilă.
3. Inițiatorul (transmițătorul) nu știa niciodată dacă mesajul a ajuns sau nu.
4. Dacă cineva avea în plan să plece în călătorie de afaceri pentru mai multe săptămâni și dorea ca toată poșta primită în acest timp să fie preluată de către secretară, acest lucru nu era ușor de realizat.
5. Interfața cu utilizatorul era slab integrată cu sistemul de transmisie, cerând utilizatorilor ca întâi să editeze un fișier, apoi să părăsească editorul și să apeleze programul de transfer de fișiere.
6. Nu era posibilă transmiterea de mesaje care să conțină o combinație de text, desene, facsimil și voce.

Pe măsură ce s-a câștigat experiență, au fost propuse sisteme de poștă electronică mai complicate. În 1982 au fost publicate propunerile cu privire la e-mail ale ARPANET, sub numele de RFC 821 (protocolul de transmisie) și RFC 822 (formatul mesajelor). Revizii minore, RFC 2821 și RFC 2822, au devenit standarde Internet, totuși toată lumea se referă la e-mail gândindu-se la RFC 822.

În 1984, CCITT a emis recomandarea X.400. După două decenii de competiție, sistemele de poștă electronică bazate pe RFC 822 sunt larg răspândite, în timp ce acelea bazate pe X.400 au dispărut. Modul în care un sistem încropit de o mână de absolvenți de știința calculatoarelor a învins un standard internațional oficial, puternic susținut de către toate PTT-urile din lumea întreagă, de multe guverne și de o parte substanțială a industriei calculatoarelor, ne aduce în minte povestea biblică a lui David și Goliat.

Motivul succesului lui RFC822 nu este dat de faptul că ar fi atât de bun, ci acela că X.400 a fost atât de slab proiectat și atât de complex, încât nimeni nu l-ar putea implementa bine. Având de ales între un sistem nesofisticat, dar care funcționează, cum este cel bazat pe RFC822 și sistemul de e-mail X.400, presupus cu adevărat minunat, dar nefuncțional, majoritatea organizațiilor l-au ales pe primul. Poate că este și o lecție în spatele acestei povești. De acea discuția noastră referitoare la e-mail se va concentra asupra sistemului de e-mail din Internet.

7.2.1 Arhitectură și servicii

În această secțiune vom furniza o prezentare de ansamblu a ceea ce pot face sistemele de poștă electronică și cum sunt ele organizate. Aceste sisteme constau de obicei din două subsisteme: **agenții-utilizator**, care permit utilizatorilor să citească și să trimită scrisori prin poșta electronică și **agenții de transfer de mesaje**, care transportă mesajele de la sursă la destinație. Agenții-utilizator sunt programe locale, care furnizează o metodă de a interacționa cu sistemul de e-mail bazată pe comenzi, meniuri sau grafică. Agenții de transfer de mesaje sunt, de regulă, **demoni** de sistem, adică procese care se execută în fundal. Sarcina lor este să transfere mesajele prin sistem.

În general, sistemele de poștă electronică pun la dispoziție cinci funcții de bază. Să aruncăm o privire asupra lor.

Compunerea se referă la procesul de creare a mesajelor și a răspunsurilor. Deși pentru corpul mesajului poate fi folosit orice editor de texte, sistemul însuși poate acorda asistență la adresare și la

completarea numeroaselor câmpuri antet atașate fiecărui mesaj. De exemplu, când se răspunde la un mesaj, sistemul poate extrage adresa inițiatorului din mesajul primit și o poate insera automat în locul potrivit din cadrul răspunsului.

Transferul se referă la deplasarea mesajului de la autor la receptor. În mare, aceasta necesită stabilirea unei conexiuni la destinație, sau la o mașină intermediară, emiterea mesajului și eliberarea conexiunii. Sistemul de poștă ar trebui să facă acest lucru singur, fără a deranja utilizatorul.

Raportarea se referă la informarea inițiatorului despre ce s-a întâmplat cu mesajul. A fost livrat? A fost respins? A fost pierdut? Există numeroase aplicații în care confirmarea livrării este importantă și poate avea chiar semnificație juridică. („Știți, domnule judecător, sistemul meu de poștă electronică nu e foarte de încredere, așa că presupun că citația electronică s-a pierdut pe undeva.")

Afișarea mesajelor primite este necesară pentru ca utilizatorii să-și poată citi poșta. Uneori sunt necesare conversii sau trebuie apelat un program de vizualizare special; de exemplu, dacă mesajul este un fișier PostScript, sau voce digitizată. Se mai încearcă uneori și conversii simple și formatări.

Dispoziția este pasul final și se referă la ceea ce face receptorul cu mesajul, după ce l-a primit. Posibilitățile includ eliminarea sa înainte de a-l citi, aruncarea sa după citire, salvarea sa ș.a.m.d. Ar trebui de asemenea să fie posibilă regăsirea și recitirea de mesaje deja salvate, trimiterea lor mai departe, sau procesarea lor în alte moduri.

În plus față de aceste servicii de bază, unele sisteme de e-mail, în special cele interne companiilor, dispun de o gamă variată de facilități avansate. Să menționăm pe scurt câteva dintre ele. Când utilizatorii se deplasează sau când sunt plecați pentru o perioadă de timp, pot dori ca poșta lor să fie trimisă acolo unde se găsesc, așa că sistemul ar trebui să fie capabil să facă acest lucru automat.

Majoritatea sistemelor permit utilizatorilor să-și creeze **cutii poștale (mailboxes)** pentru a păstra mesajele sosite. Sunt necesare comenzi de creare și distrugere a cutiilor poștale, de inspectare a conținutului acestora, de inserare și de ștergere de mesaje din cutii poștale ș.a.m.d.

Managerii de companii au adesea nevoie să trimită un același mesaj fiecărui subordonat, client sau furnizor. Acest lucru dă naștere ideii de **listă de poștă (mailing list)**, care este o listă de adrese de poștă electronică. Când un mesaj este trimis la lista de poștă, copii identice ale sale sunt expediate fiecăruia dintre cei de pe listă.

Alte caracteristici evoluate sunt copii la indigo, poștă de prioritate mare, poștă secretă (criptată), receptori alternativi, dacă cel primar nu este disponibil, și posibilitatea de a permite secretarelor să se ocupe de poșta primită de șefii lor.

Poșta electronică este în prezent folosită pe scară largă în industrie, pentru comunicație în cadrul companiilor. Aceasta permite unor angajați, răspândiți la distanțe mari unii de ceilalți, chiar și peste mai multe fusuri orare, să coopereze la proiecte complexe. Eliminând majoritatea indiciilor cu privire la funcție, vârstă și gen, dezbaterile prin poșta electronică tind să se concentreze asupra ideilor și nu a statutului din cadrul organizației. Prin poșta electronică, o idee sclipitoare a unui student la cursurile de vară poate avea un impact mai mare decât una stupidă, venită de la un vicepreședinte executiv.

O idee fundamentală în toate sistemele moderne de e-mail este distincția dintre **plic** și conținutul său. Plicul încapsulează mesajul. Conține toată informația necesară pentru transportul mesajului, cum ar fi destinația, adresa, prioritatea, nivelul de securitate, toate acestea fiind distincte de mesajul în sine. Agenții de transfer de mesaje folosesc plicul pentru rutare (dirijare), așa cum face și oficiul poștal.

Mesajul din interiorul plicului conține două părți: **antetul** și **corpul**. Antetul conține informație de control pentru agenții utilizator. Corpul mesajului se adresează în întregime utilizatorului uman. Plicurile și mesajele sunt ilustrate în fig. 7-7.

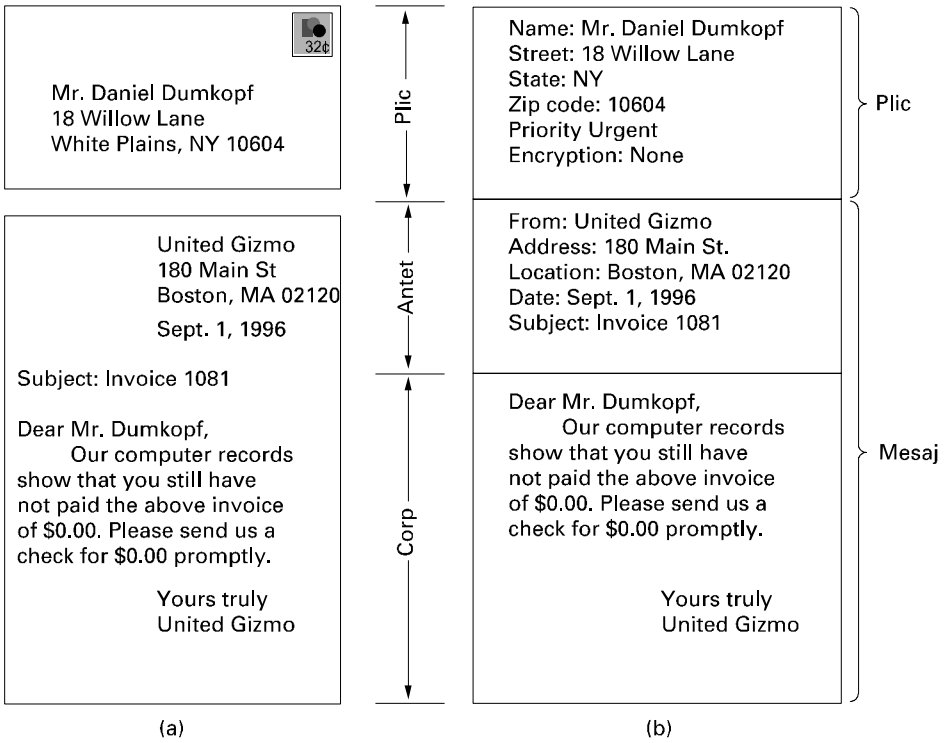


Fig. 7-7. Plicuri și mesaje. (a) poștă clasică (b) poștă electronică.

7.2.2 Agentul utilizator

Sistemele de poștă electronică au, așa cum am văzut, două părți esențiale: agenții-utilizator și agenții de transfer de mesaje. În această secțiune ne vom uita la agenții utilizator. Un agent utilizator este de obicei un program (uneori numit cititor de poștă) care acceptă o varietate de comenzi pentru compunerea, primirea și răspunsul la mesaje, cât și pentru manipularea cutiilor poștale. Unii agenți-utilizator au o interfață sofisticată, dirijată prin meniuri sau icoane, care necesită un maus, în timp ce altele acceptă comenzi de câte un caracter, date de la tastatură. Funcțional însă, toți aceștia sunt identici. Unele sisteme sunt dirijate prin meniuri sau icoane dau au și alternative mai „scurte" pe tastatură.

Trimiterea poștei electronice

Pentru a trimite un mesaj prin poșta electronică, un utilizator trebuie să furnizeze mesajul, adresa destinație, și eventual alți câțiva parametri. Mesajul poate fi produs cu un editor de texte de sinestătător, cu un program de procesare de text sau, eventual, cu un editor de texte specializat, construit în interiorul agentului utilizator. Adresa de destinație trebuie să fie într-un format cu care agentul utilizator să poată lucra. Mulți agenți-utilizator solicită adrese de forma *utilizator@adresă-dns.* Deoarece aceste lucruri au fost studiate anterior în acest capitol, nu vom relua materialul respectiv aici.

Oricum, merită notat că există și alte forme de adresare. În particular, adresele X.400 arată radical diferit de cele DNS. Ele sunt compuse din perechi de forma *atribut = valoare*, separate de bare oblice. De exemplu:

/C=US/SP=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Această adresă specifică o țară, un stat, o localitate, o adresă personală și un nume obișnuit (Ken Smith). Sunt posibile multe alte atribute, astfel încât poți trimite mesaje cuiva al cărui nume nu-l știi, atâta timp cât știi suficiente alte atribute (de exemplu, compania și funcția). Cu toate că adresele X.400 sunt mult mai puțin convenabile decât cele DNS, cele mai multe sisteme de poștă electronică permit folosirea de **pseudonime** (**aliases**, uneori numite și porecle) pentru obținerea numelor sau adreselor de e-mail corecte ale unei persoane. În consecință, chiar și cu adresele de tip X.400, de obicei nu este necesară scrierea în întregime a acelor șiruri ciudate.

Majoritatea sistemelor de e-mail acceptă liste de poștă, astfel că un utilizator poate trimite, cu o singură comandă, un același mesaj tuturor persoanelor dintr-o listă. Dacă lista de poștă este păstrată local, agentul-utilizator poate pur și simplu să trimită câte un mesaj separat fiecăruia dintre receptorii doriți. Dacă lista este păstrată la distanță, atunci mesajele vor fi expandate acolo. De exemplu, dacă un grup de admiratori de păsări au o listă de poștă numită *birders*, instalată la *meadowlark.arizona.edu*, atunci orice mesaj trimis la *birders@meadowlark.arizona.edu* va fi dirijat către Universitatea din Arizona și expandat acolo în mesaje individuale pentru toți membrii listei de poștă, oriunde ar fi ei în lume. Utilizatorii acestei liste de poștă nu pot determina că aceasta este o listă de adrese. Ar putea fi la fel de bine cutia poștală personală a Prof. Gabriel O. Birders.

Citirea poștei electronice

În mod obișnuit, când este lansat un agent-utilizator, înainte de a afișa ceva pe ecran, el se va uita în cutia poștală a utilizatorului după mesajele care sosesc. Apoi poate anunța numărul de mesaje din cutie, sau poate afișa pentru fiecare mesaj câte un rezumat de o linie, pentru ca apoi să aștepte o comandă.

Ca exemplu despre cum lucrează un agent-utilizator, să aruncăm o privire asupra unui scenariu tipic pentru poșta electronică. După lansarea agentului-utilizator, utilizatorul cere un rezumat al mesajelor sale. O imagine ca aceea din fig. 7-8 apare în acest caz pe ecran. Fiecare linie se referă la câte un mesaj. În acest exemplu, cutia poștală conține opt mesaje.

#	Marcaje	Octeți	Transmițător	Subiect
1	K	1030	Asw	Changes to MINIX
2	KA	6348	Trudy	Not all Trudys are nasty
3	K F	4519	Amy N. Wong	Request for information
4		1236	Bal	Bioinformatics
5		103610	Kaashoek	Material on peer-to-peer
6		1223	Frank	Re: Will you review a grant proposal
7		3110	Guido	Our paper has been accepted
8		1204	Dmr	Re: My student's visit

Fig. 7-8. Un exemplu de afișare a conținutului unei cutii poștale.

Fiecare linie de afișaj conține câteva câmpuri extrase de pe plicul sau din antetul mesajului corespunzător. Într-un sistem simplu de poștă electronică, alegerea câmpurilor afișate este făcută în cadrul programului. Într-un sistem mai sofisticat, utilizatorul poate specifica ce câmpuri să fie afișate, furnizând un **profil al utilizatorului**, adică un fișier care descrie formatul de afișare. În exemplul

considerat, primul câmp reprezintă numărul mesajului. Al doilea câmp, *Marcaje*, poate conține un *K*, însemnând că mesajul nu este nou, dar a fost citit anterior și păstrat în cutia poștală; un *A*, însemnând că deja s-a răspuns la acest mesaj; și/sau un *F*, însemnând că mesajul a fost trimis mai departe altcuiva. Sunt de asemenea posibile și alte marcaje.

Al treilea câmp specifică lungimea mesajului și al patrulea spune cine a trimis mesajul. Din moment ce el este pur și simplu extras din mesaj, acest câmp poate conține prenume, nume complete, inițiale, nume de cont, sau orice altceva și-a ales transmițătorul să pună. În sfârșit, câmpul *Subiect* specifică despre ce este mesajul, într-un scurt rezumat. Persoanele care omit să includă un câmp *Subiect* adesea descoperă că răspunsurile la scrisorile lor tind să nu obțină prioritate maximă.

După ce au fost afișate antetele, utilizatorul poate executa oricare dintre comenzile disponibile, ca de exemplu afișarea unui mesaj, ștergerea unui mesaj și așa mai departe. Sistemele mai vechi lucrau în mod text și de obicei foloseau comenzi de un caracter pentru diversele operații, ca de exemplu T (scrie mesaj), A (răspunde la mesaj), D (șterge mesaj) și F (trimite mai departe). Un argument specifica mesajul corespunzător. Sistemele mai recente folosesc interfețe grafice. De obicei, utilizatorul selectează un mesaj cu mouse-ul și apoi apasă pe o iconiță pentru a scrie, răspunde la mesaj, sau pentru a-l trimite mai departe.

Poșta electronică a parcurs un drum lung de pe vremea când era doar transfer de fișiere. Agenții-utilizator sofisticați fac posibilă manevrarea unui volum mare de scrisori. Pentru persoane care primesc și trimit mii de mesaje pe an, asemenea instrumente sunt neprețuite.

7.2.3 Formatele mesajelor

Să ne întoarcem acum de la interfața utilizator la formatul mesajelor de poștă electronică în sine. Mai întâi ne vom uita la e-mailul ASCII de bază, care utilizează RFC 822. După aceea ne vom concentra asupra extensiilor multimedia ale RFC 822.

RFC 822

Mesajele constau dintr-un plic simplu (descris în RFC 821), un număr de câmpuri antet, o linie goală și apoi corpul mesajului. Fiecare câmp antet se compune (din punct de vedere logic) dintr-o singură linie de text ASCII, conținând numele câmpului, două puncte, și, pentru majoritatea câmpurilor, o valoare. RFC 822 a fost creat acum două decenii și nu distinge clar plicul de câmpurile antet, cum ar face un standard nou. Cu toate că a fost corectat în RFC 2822, o refacere completă n-a fost posibilă datorită răspândirii sale largi. La o utilizare normală, agentul-utilizator construiește un mesaj și îl transmite agentului de transfer de mesaje, care apoi folosește unele dintre câmpurile antet pentru a construi plicul efectiv, o combinație oarecum demodată de mesaj și plic.

Principalele câmpuri antet, legate de transportul de mesaje, sunt înfățișate în fig. 7-9. Câmpul *To:* oferă adresa DNS a receptorului primar. Este permisă de asemenea existența de receptori multipli. Câmpul *Cc:* dă adresa oricărui receptor secundar. În termenii livrării, nu este nici o diferență între un receptor primar și unul secundar. Este în întregime o deosebire psihologică, ce poate fi importantă pentru persoanele implicate, dar este neimportantă pentru sistemul de poștă. Termenul *Cc:* (Carbon copy - copie la indigo) este puțin depășit, din moment ce calculatoarele nu folosesc indigo, dar este bine înrădăcinat. Câmpul *Bcc:* (Blind carbon copy - copie confidențială la indigo) este la fel ca *Cc:*, cu excepția că această linie este ștearsă din toate copiile trimise la receptorii primari și secundari. Acest element permite utilizatorilor să trimită copii unei a treia categorii de receptori, fără ca cei primari și secundari să știe acest lucru.

Antet	Conținut
To:	Adresa(ele) de e-mail a(le) receptorului(ilor) primar(i)
Cc:	Adresa(ele) de e-mail a(le) receptorului(ilor) secundar(i)
Bcc:	Adresa(ele) de e-mail pentru „blind carbon copy"
From:	Persoana sau persoanele care au creat mesajul
Sender:	Adresa de e-mail a transmițătorului curent
Received:	Linie adăugată de fiecare agent de transfer de-a lungul traseului
Return-Path:	Poate fi folosită pentru a identifica o cale de întoarcere la transmițător

Fig. 7-9. Câmpurile antet ale lui RFC 822, legate de transportul de mesaje.

Următoarele două câmpuri, *From:* și *Sender:*, precizează cine a scris și respectiv cine a trimis mesajul. Acestea pot să nu fie identice. De exemplu, se poate ca o directoare executivă să scrie un mesaj, dar ca secretara ei să fie cea care îl trimite efectiv. În acest caz, directoarea executivă va fi afișată în câmpul *From:* și secretara în câmpul *Sender:*. Câmpul *From:* este obligatoriu, dar câmpul *Sender:* poate fi omis dacă este identic cu *From:*. Aceste câmpuri sunt necesare în cazul în care mesajul nu poate fi livrat și trebuie returnat transmițătorului.

O linie conținând *Received:* este adăugată de fiecare agent de transfer de mesaje de pe traseu. Linia conține identitatea agentului, data și momentul de timp la care a fost primit mesajul și alte informații care pot fi utilizate pentru găsirea defecțiunilor în sistemul de dirijare.

Câmpul *Return-Path:* este adăugat de agentul final de transfer de mesaje și are în intenție să indice cum se ajunge înapoi la transmițător. În teorie, această informație poate fi adunată din toate antetele *Received:* (cu excepția numelui cutiei poștale a transmițătorului), dar rareori este completată așa și de obicei conține chiar adresa transmițătorului.

Antet	Conținut
Date:	Data și momentul de timp la care a fost trimis mesajul
Reply-To:	Adresa de e-mail la care ar trebui trimise răspunsurile
Message-Id:	Număr unic, utilizat ulterior ca referință pentru acest mesaj (identificator)
In-Reply-To:	Identificatorul mesajului al cărui răspuns este mesajul curent
References:	Alți identificatori de mesaje relevanți
Keywords:	Cuvinte cheie alese de utilizator
Subject:	Scurt cuprins al mesajului, afișabil pe o singură linie

Fig. 7-10. Câteva câmpuri utilizate în antetul lui RFC 822.

În plus față de câmpurile din fig. 7-9, mesajele RFC 822 pot conține de asemenea o varietate de câmpuri antet, folosite de agenții-utilizator sau de receptorii umani. Cele mai des întâlnite dintre ele sunt prezentate în fig. 7-10. Majoritatea lor se explică de la sine, deci nu vom intra în detaliu la toate.

Câmpul *Reply-To:* este uneori utilizat când nici persoana care a compus mesajul, nici cea care l-a trimis nu vrea să vadă răspunsul. De exemplu, un director de marketing scrie un mesaj prin e-mail pentru a spune clienților despre un nou produs. Mesajul este trimis de o secretară, dar câmpul *Reply-To:* conține șeful departamentului de vânzări, care poate răspunde la întrebări și primi comenzi. Acest câmp este foarte folositor când transmițătorul are două conturi de e-mail și vrea ca răspunsul să ajungă în celălalt.

Documentul RFC 822 afirmă explicit că utilizatorilor le este permis să inventeze noi antete, atâta timp cât acestea încep cu șirul de caractere X-. Se garantează că nici o extindere ulterioară nu va folosi nume ce încep cu X-, pentru a evita conflictele (suprapunerile) dintre antetele oficiale și cele

personale. Uneori studenții care fac pe deștepții includ câmpuri de tipul *X-Fruit-of-the-Day:* sau *X-Disease-of-the-Week:*, care sunt legale, deși nu întotdeauna clarificatoare.

După antete urmează corpul mesajului. Aici utilizatorii pot pune orice vor. Unii oameni își încheie mesajele cu semnături elaborate, incluzând caricaturi ASCII simple, citate din personalități mai mari sau mai mici, declarații politice și declinări de tot felul (de exemplu: Corporația XYZ nu este răspunzătoare pentru părerile mele; de fapt nu poate nici măcar să le înțeleagă).

MIME - Multipurpose Internet Mail Extensions (extensii de poștă cu scop multiplu)

La începuturile ARPANET, poșta electronică consta exclusiv din mesaje de tip text, scrise în engleză și exprimate în ASCII. Pentru acest context, RFC 822 realiza sarcina complet: specifică antetele, dar lăsa conținutul în întregime în seama utilizatorilor. În zilele noastre, această abordare nu mai este adecvată pentru Internetul care se întinde în lumea întreagă. Problemele includ transmisia și recepția de:

1. Mesaje în limbi cu accente (de exemplu franceza și germana).
2. Mesaje în alfabete ne-latine (de exemplu ebraică și rusă).
3. Mesaje în limbi fără alfabet (de exemplu chineză și japoneză).
4. Mesaje care nu conțin text deloc (de exemplu audio și video).

O soluție posibilă a fost propusă în RFC 1341 și actualizată în RFC-urile 2045-2049. Această soluție, numită **MIME (Multipurpose Internet Mail Extensions),** este în prezent larg utilizată. O vom descrie în continuare. Pentru informații suplimentare în legătură cu MIME, vedeți RFC-urile.

Ideea fundamentală a MIME este să continue să folosească formatul RFC 822, dar să adauge structură corpului mesajului și să definească reguli de codificare pentru mesajele non-ASCII. Deoarece respectă RFC 822, mesajele MIME pot fi trimise utilizând programele și protocoalele de poștă existente. Tot ceea ce trebuie modificat sunt programele de transmitere și recepție, pe care utilizatorii le pot face ei înșiși.

Antet	Conținut
MIME-Version:	Identifică versiunea de MIME
Content-Description:	Șir adresat utilizatorului care spune ce este în mesaj
Content-Id:	Identificator unic
Content-Transfer-Encoding:	Cum este împachetat corpul pentru transmisie
Content-Type:	Natura mesajului

Fig. 7-11. Antetele RFC 822 adăugate de către MIME.

MIME definește cinci noi antete de mesaje, așa cum se arată în fig. 7-11. Primul dintre acestea specifică pur și simplu agentului-utilizator care primește mesajul că este vorba de un mesaj MIME și ce versiune de MIME utilizează. Orice mesaj care nu conține un antet *MIME-Version:* este presupus ca fiind un mesaj în text pur, în engleză, și este procesat ca atare.

Antetul *Content-Description:* este un șir de caractere ASCII specificând ce este în mesaj. Acest antet este necesar pentru ca receptorul să știe dacă merită să decodifice și să citească mesajul. Dacă șirul de caractere spune: " Fotografia hamsterului Barbarei" și persoana care primește mesajul nu este un mare iubitor de hamsteri, mesajul va fi probabil mai curând aruncat, decât decodificat într-o fotografie color de înaltă rezoluție.

Antetul *Content-Id:* identifică conținutul. Utilizează același format ca antetul standard *Message-Id:*.

Antetul *Content-Transfer-Encoding:* arată cum este împachetat pentru transmisie corpul mesajului, într-o rețea care poate ridica obiecții la majoritatea caracterelor diferite de litere, cifre și semne de punctuație. Sunt furnizate cinci scheme (plus o evadare către noi scheme). Cea mai simplă schemă se referă chiar la text ASCII. Caracterele ASCII utilizează 7 biți și pot fi transportate direct prin protocolul de e-mail, atâta timp cât nici o linie nu are mai mult de 1000 de caractere.

Următoarea schemă ca simplitate este cam același lucru, dar utilizează caractere de câte 8 biți, reprezentând toate valorile de la 0 la 255 inclusiv. Această schemă de codificare încalcă protocolul (original) de e-mail utilizat în Internet, dar este folosită de unele părți ale Internetului, care implementează niște extensii ale protocolului original. În timp ce declararea codificării nu o face să devină legală, faptul că o avem explicit poate cel puțin să lămurească lucrurile atunci când ceva merge prost. Mesajele utilizând codificarea de 8 biți trebuie încă să respecte lungimea maximă a liniei, care este standard.

Este chiar mai rău în cazul mesajelor care utilizează codificare binară. Aceste mesaje sunt fișiere binare arbitrare, care nu numai că utilizează toți cei 8 biți, dar nu respectă nici limita de linie de 1000 de caractere. Programele executabile intră în această categorie. Nu se acordă nici o garanție că mesaje binare vor ajunge corect, dar mulți le trimit oricum.

Modalitatea corectă de a codifica mesaje binare este de a utiliza **codificarea în bază 64**, numită uneori **armură ASCII.** În această schemă, grupuri de câte 24 de biți sunt împărțite în patru unități de câte 6 biți, fiecare dintre aceste unități fiind transmisă ca un caracter ASCII legal. Codificarea este „A” pentru 0, „B” pentru 1, ș.a.m.d., urmate de cele 26 de litere mici, cele 10 cifre, și în cele din urmă + și / pentru 62 și respectiv 63. Secvențele == și = sunt utilizate pentru a arăta că ultimul grup a conținut doar 8 sau respectiv 16 biți. Se ignoră secvențele carriage return și line feed, astfel că ele pot fi inserate după dorință, pentru a păstra liniile suficient de scurte. Utilizând această schemă pot fi trimise sigur texte binare arbitrare.

Pentru mesajele care sunt aproape în întregime ASCII și conțin puține caractere ne-ASCII, codificarea în bază 64 este oarecum ineficientă. În locul acesteia se utilizează o codificare numită **quoted-printable-encoding** (codificare afișabilă marcată). Aceasta este o codificare de tip ASCII pe 7 biți, având toate caracterele cu cod mai mare de 127 codificate sub forma unui semn egal urmat de valoarea caracterului reprezentată prin două cifre hexazecimale.

Rezumând, datele binare ar trebui trimise codificate în bază 64 sau sub formă quoted-printable. Când există motive întemeiate pentru a nu utiliza una dintre aceste scheme, este posibil să se specifice în antetul Content-Transfer-Encoding: o codificare definită de către utilizator.

Ultimul antet înfățișat în fig. 7-11 este cu adevărat cel mai interesant. El specifică natura corpului mesajului. În RFC 2045 sunt definite șapte tipuri, fiecare având unul sau mai multe subtipuri. Tipul și subtipul sunt separate printr-o bară oblică (slash), ca în:

```
Content-Type: video/mpeg
```

Subtipul trebuie precizat explicit în antet; nu sunt furnizate valori implicite. Lista inițială de tipuri și subtipuri specificate în RFC 2045 este prezentată în fig. 7-12. De atunci au fost adăugate multe altele, introducându-se intrări adiționale de fiecare dată când a devenit necesar.

Să parcurgem acum lista tipurilor. Tipul *text* este utilizat pentru text simplu. Combinația *text/plain* este folosită pentru mesaje obișnuite care pot fi afișate de îndată ce sunt primite, fără codificare sau procesare ulterioară. Această opțiune permite ca mesajele obișnuite să fie transportate în MIME adăugând doar câteva antete suplimentare.

Tip	Subtip	Descriere
Text	Plain	Text neformatat
	Enriched	Text incluzând comenzi simple de formatare
Image	Gif	Imagini fixe în format GIF
	Jpeg	Imagini fixe în format JPEG
Audio	Basic	Sunet
Video	Mpeg	Film în format MPEG
Application	Octet-stream	Secvență neinterpretată de octeți
	Postscript	Un document afișabil în PostScript
Message	Rfc822	Un mesaj MIME RFC 822
	Partial	Mesajul a fost fragmentat pentru transmisie
	External-body	Mesajul în sine trebuie adus din rețea
Multipart	Mixed	Părți independente în ordine specificată
	Alternative	Același mesaj în formate diferite
	Parallel	Părțile trebuie vizualizate simultan
	Digest	Fiecare parte este un mesaj RFC 822 complet

Fig. 7-12. Tipurile și subtipurile aparținând MIME definite în RFC 2045.

Subtipul *text/enriched* permite includerea în text a unui limbaj simplu de marcare. Acest limbaj furnizează o modalitate independentă de sistem pentru a exprima scrierea cu caractere aldine sau cursive, dimensiunile, alinierea, distanțele dintre rânduri, folosirea de indici superiori sau inferiori și paginarea simplă. Limbajul de marcare se bazează pe SGML, Standard Generalized Markup Language (limbajul standard generalizat de marcare), folosit de asemenea ca bază pentru HTML, utilizat în World Wide Web. De exemplu mesajul

The <bold> time </bold> has come the <italic> walrus </italic> said ...

ar fi afișat sub forma:

The **time** has come the *walrus* said...

Depinde de sistemul receptor să aleagă interpretarea potrivită. Dacă sunt disponibile caractere aldine și cursive, acestea vor putea fi folosite; altfel, pentru a scoate în evidență se pot utiliza culori, scriere cu clipire sau video-invers etc. Sisteme diferite pot face alegeri diferite.

Când Web-ul a devenit popular, a fost adăugat un nou subtip, *text/html* (în RFC 2854) pentru a permite paginilor Web să fie trimise într-un e-mail de tip RFC 822. Un subtip pentru sistemul extins de marcare, *text/xml*, este definit in RFC 3023. Vom studia HTML și XML mai târziu în acest capitol.

Următorul tip MIME este *image*, utilizat pentru trimiterea de imagini fixe. În zilele noastre sunt utilizate multe formate, atât cu, cât și fără compresie, pentru a păstra și transmite imagini. Două dintre acestea, GIF și JPEG, sunt recunoscute de aproape toate programele de navigare, dar există și altele care au fost adăugate la lista originală.

Tipurile *video* și *audio* sunt pentru imagini în mișcare și respectiv pentru imagini cărora li se asociază și sunet. Trebuie notat că *video* include doar informația video, nu și coloana sonoră. Dacă trebuie transmis un film cu sunet, s-ar putea ca porțiunile audio și video să trebuiască să fie transmise separat, depinzând de sistemul de codificare utilizat. Primul format video definit a fost cel inventat de cei ce se intitulează modest Moving Picture Experts Group (MPEG - Grupul de experți în imagini în mișcare), dar de atunci au fost adăugate și altele. În plus față de *audio/basic*, un nou tip audio, *audio/mpeg* a fost adăugat în RFC 3003 pentru a permite oamenilor să trimită fișiere MP3 prin e-mail.

Tipul *application* este utilizat ca un colector pentru formatele care necesită prelucrare externă, neidentificate de nici unul dintre celelalte tipuri. Un *octet-stream* este doar o secvență de octeți nein-

terpretați. La primirea unui asemenea flux, un agent-utilizator ar trebui probabil să-l afișeze, sugerându-i utilizatorului să-l copieze într-un fișier și cerându-i un nume pentru acesta. Procesarea ulterioară este apoi la latitudinea utilizatorului.

Celălalt subtip definit este *postscript*, care se referă la limbajul PostScript, produs de Adobe Systems și larg utilizat pentru descrierea paginilor imprimate. Multe imprimante au înglobate interpretoare PostScript. Deși un agent-utilizator poate pur și simplu să apeleze un interpretor PostScript extern pentru a interpreta fișierele PostScript primite, acest lucru nu este lipsit de pericole. PostScript este un întreg limbaj de programare. Dându-i-se destul timp, o persoană suficient de masochistă ar putea scrie în PostScript un compilator de C, sau un sistem de management de baze de date. Afișarea unui mesaj primit în format PostScript se face executând programul PostScript conținut de acesta. Pe lângă afișarea unui text, acest program poate citi, modifica, sau șterge fișierele utilizatorului și poate avea și alte efecte laterale neplăcute.

Tipul *message* permite încapsularea în întregime a unui mesaj în altul. Această schemă este utilă, de exemplu pentru trimiterea mai departe a e-mailului, cu *forward*. Când un mesaj RFC 822 complet este încapsulat într-un mesaj exterior, ar trebui utilizat subtipul *rfc822.*

Subtipul *partial* face posibilă împărțirea unui mesaj încapsulat în bucăți de mesaj și trimiterea separată a acestora (de exemplu, dacă mesajul încapsulat este prea lung). Parametrii fac posibilă reasamblarea în ordinea corectă a tuturor părților, la destinație.

Și în sfârșit, subtipul *external-body* pate fi utilizat pentru mesaje foarte lungi (de exemplu, filme video). În loc de a include fișierul MPEG în mesaj, se dă o adresă FTP și agentul utilizator al receptorului o poate aduce din rețea în momentul în care este necesar. Această facilitate este în special utilă când se trimite un film la o întreagă listă de poștă și se presupune că doar câțiva dintre membrii acesteia îl vor vedea (gândiți-vă la e-mailurile inutile, conținând reclame video).

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Pământul înconjoară soarele de un număr întreg de ori

Acesta este preambulul. Agentul utilizator îl ignoră. O zi bună.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/richtext

Happy birthday to you
Happy birthday to you
Happy birthday dear <bold> Carolyn </bold>
Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
 access-type=„anon-ftp”;
 site=„bicycle.abcd.com”;
 directory=„pub”;
 name=„birthday.snd;

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm
```

Fig. 7-13. Un mesaj multipart conținând alternative de tip text formatat și audio.

Ultimul tip este *multipart*, care permite unui mesaj să conțină mai multe părți, începutul și sfârșitul fiecărei părți fiind clar delimitat. Subtipul *mixed* permite fiecărei părți să fie diferită de celelalte, fără a avea o structură adițională impusă. Multe programe de e-mail permit utilizatorului să aibă una sau mai multe părți atașate la un mesaj text. Acestea sunt trimise folosind tipul *multipart*.

În contrast cu tipul *multipart*, subtipul *alternative* permite ca fiecare parte să conțină același mesaj, dar exprimat într-un alt mediu sau într-o codificare diferită. De exemplu, un mesaj ar putea fi trimis ca ASCII simplu, ca text formatat și ca PostScript. Un agent-utilizator proiectat corespunzător, la primirea unui asemenea mesaj, îl va afișa, dacă va fi posibil, în PostScript. A doua alegere va fi textul formatat. Dacă nici una dintre aceste alternative nu ar fi posibilă, s-ar afișa text ASCII obișnuit. Părțile ar trebui ordonate de la cea mai simplă, la cea mai complexă, pentru a ajuta receptorii care folosesc agenți-utilizator pre-MIME să înțeleagă mesajul (chiar și un utilizator pre-MIME poate citi text ASCII simplu). Subtipul *alternative* poate fi folosit de asemenea pentru limbaje multiple. În acest context, Rosetta Stone poate fi privit ca precursor al mesajului de tip *multipart/alternative*.

Un exemplu multimedia este prezentat în fig. 7-13. Aici, o felicitare este transmisă atât sub formă de text cât și sub formă de cântec. Dacă receptorul are facilități audio, agentul utilizator va aduce fișierul de sunet, *birthday.snd* și îl va interpreta. Dacă nu, versurile vor fi afișate pe ecran într-o liniște de mormânt. Părțile sunt delimitate de două cratime urmate de șirul (definit de utilizator) specificat în parametrul *boundary.*

Observați că antetul *Content-Type* apare în trei poziții în acest exemplu. La primul nivel indică faptul că mesajul are mai multe părți. În cadrul fiecărei părți specifică tipul și subtipul acesteia. În sfârșit, în corpul celei de-a doua părți, este necesar pentru a indica agentului utilizator ce fel de fișier extern trebuie să aducă. Pentru a exprima ușoara diferență de utilizare, s-au folosit litere mici, deși toate antetele sunt *case insensitive* (nu fac diferență între literele mari și cele mici). Antetul *content-transfer-encoding* este în mod similar necesar pentru orice corp extern care nu este codificat ca ASCII pe 7 biți.

Întorcându-ne la subtipurile corespunzătoare mesajelor *multipart*, vom spune că mai există două posibilități. Subtipul *parallel* este utilizat când toate părțile trebuie să fie interpretate simultan. De exemplu, adesea filmele au un canal audio și unul video. Ele sunt mai de efect dacă aceste două canale sunt interpretate în paralel și nu consecutiv.

În sfârșit, subtipul *digest* este utilizat când multe mesaje sunt împachetate împreună, într-unul compus. De exemplu, niște grupuri de dialog de pe Internet pot aduna mesaje de la abonații lor și apoi să le trimită în afară ca un singur mesaj de tip *multipart/digest*.

7.2.4 Transferul mesajelor

Sistemul de transfer de mesaje se ocupă cu transmiterea mesajelor de la expeditor la receptor. Cea mai simplă cale de a realiza acest lucru constă în stabilirea unei conexiuni de transport de la mașina sursă la cea de destinație și apoi, pur și simplu în trimiterea mesajului. După ce examinăm cum se face acest lucru în mod normal, vom studia câteva situații în care metoda nu funcționează și vom vedea ce trebuie făcut în aceste cazuri.

SMTP – Simple Mail Transfer Protocol (Protocol simplu de transfer de poștă)

În cadrul Internetului poșta electronică este livrată prin stabilirea de către mașina sursă a unei conexiuni TCP la portul 25 al mașinii de destinație. La acest port se află un demon de e-mail care știe **SMTP (Simple Mail Transfer Protocol)**. Acest demon acceptă conexiunile și copiază mesajele

de la ele în cutiile poștale corespunzătoare. Dacă mesajul nu poate fi livrat, se returnează transmițătorului un raport de eroare conținând prima parte a mesajului nelivrat.

SMTP este un protocol simplu de tip ASCII. După stabilirea conexiunii TCP la portul 25, mașina transmițătoare, operând în calitate de client, așteaptă ca mașina receptoare, operând ca server, să vorbească prima. Serverul începe prin a trimite o linie de text, declarându-și identitatea și spunând dacă este pregătit sau nu să primească mesaje. Dacă nu este, clienții eliberează conexiunea și încearcă din nou mai târziu.

```
                S: 220 xyz.com SMTP service ready
C: HELO abcd.com
                S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
                S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
                S: 250 recipient ok
C: DATA
        S: 354 Trimite mail; terminat cu "." pe linie nouă
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Pământul înconjoară soarele de un număr întreg de ori
C:
C: Acesta este preambulul. Agentul utilizator îl ignoră. O zi bună.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Carolyn </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.abcd.com";
C: directory="pub";
C: name="birthday.snd";
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                S: 250 message accepted
C:QUIT
                S: 221 xyz.com closing connection
```

Fig. 7-14. Transferul unui mesaj de la *elinor@abcd.com* la *carolyn@zxy.com*.

Dacă serverul este dispus să primească e-mail, clientul anunță de la cine vine scrisoarea și cui îi este adresată. Dacă un asemenea receptor există la destinație, serverul îi acordă clientului permisiunea să trimită mesajul. Apoi clientul trimite mesajul și serverul îl confirmă. În general nu este necesară atașarea unei sume de control deoarece TCP furnizează un flux sigur de octeți. Dacă mai există și alte mesaje, acestea sunt trimise tot acum. Când schimbul de mesaje, în ambele direcții, s-a încheiat, conexiunea este eliberată. În fig. 7-14 este prezentată o mostră de dialog referitoare la trimiterea mesajului din fig. 7-13, incluzând codurile numerice utilizate de SMTP. Liniile trimise de client sunt marcate cu C:, iar cele trimise de server cu S:.

Câteva comentarii în legătură cu fig. 7-14 ar putea fi utile. Prima comandă a clientului este într-adevăr *HELO*. Din posibilele abrevieri de patru caractere ale cuvântului *HELLO,* aceasta are numeroase avantaje față de concurenta sa cea mai mare. Motivul pentru care toate comenzile trebuiau să aibă patru caractere s-a pierdut în negura vremii.

În Fig.7-14, mesajul este trimis la un singur receptor și de aceea este utilizată o singură comandă *RCPT*. Mai multe asemenea comenzi sunt permise pentru a trimite un singur mesaj mai multor receptori. Fiecare dintre ele este confirmată sau rejectată individual. Chiar dacă unii dintre receptori sunt rejectați (deoarece ei nu există la destinație), mesajul poate fi trimis celor rămași.

În sfârșit, deși sintaxa comenzilor de patru caractere de la client este rigid specificată, sintaxa replicilor este mai puțin rigidă. Doar codul numeric contează cu adevărat. Fiecare implementare poate pune după cod ce șiruri de caractere vrea.

Pentru a înțelege mai bine cum funcționează SMTP și câteva din celelalte protocoale descrise în acest capitol, încercați-le! În orice caz, mai întâi mergeți la o mașină conectată la Internet. Într-un sistem UNIX introduceți comanda:

```
telnet mail.isp.com 25
```

înlocuind numele DNS cu numele serverului de mail al ISP-ului dvs. Pe un sistem Windows, faceți clic pe Start, apoi Run, apoi tastați comanda în căsuța de dialog. Această comandă va stabili o conexiune Telnet (adică TCP) pe portul 25 pe mașina respectivă. Portul 25 este portul SMTP (vezi Fig. 6-27 pentru câteva porturi uzuale). Probabil o sa primiți un răspuns de genul:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Primele trei linii spun ce face telnet-ul. Ultima linie este de la serverul SMTP de pe mașina de la distanță, anunțând disponibilitatea acesteia de a vorbi cu dvs. și de a accepta e-mail. Pentru a vedea ce comenzi acceptă, tastați:

```
HELP
```

De aici înainte, este posibilă o secvență de comenzi ca cea din fig. 7-14, începând cu comanda *HELO* dată de client.

E bine de notat faptul că folosirea liniilor de text ASCII pentru comenzi nu e un accident. Cele mai multe protocoale Internet funcționează așa. Folosirea textului ASCII face ca protocoalele foarte ușor de testat și depanat. Ele pot fi testate trimițând manual comenzi, cum am văzut mai sus, pentru care copiile mesajelor (eng.: dumps) sunt ușor de citit.

Chiar dacă protocolul SMTP este bine definit, mai pot apărea câteva probleme. O problemă este legată de lungimea mesajelor. Unele implementări mai vechi nu pot să lucreze cu mesaje mai mari de 64KB. O altă problemă se referă la expirări de timp (timeout). Dacă acestea diferă pentru server și client, unul din ei poate renunța, în timp ce celălalt este încă ocupat, întrerupând conexiunea în mod neașteptat. În sfârșit, în unele situații, pot fi lansate schimburi infinite de mesaje. De exemplu, dacă mașina 1 păstrează lista de poștă A și mașina 2 lista de poștă B și fiecare listă conține o intrare pentru cealaltă, atunci orice mesaj trimis oricăreia dintre cele două liste va genera o cantitate nesfârșită de trafic de e-mail.

Pentru a atinge câteva dintre aceste probleme, în RFC 2821 s-a definit protocolul SMTP extins (**ESMTP**). Clienții care doresc să-l utilizeze trebuie să trimită inițial un mesaj *EHLO* în loc de *HELO*. Dacă acesta este rejectat, atunci serverul este unul standard de tip SMTP și clientul va trebui să se comporte în modul obișnuit. Dacă *EHLO* este acceptat, înseamnă ca sunt permise noile comenzi și noii parametri.

7.2.5 Livrarea finală

Până acum, am presupus că toți utilizatorii lucrează pe mașini capabile să trimită și să primească e-mail. După cum am văzut, e-mail-ul este livrat prin stabilirea unei conexiuni TCP între expeditor și destinatar și apoi prin trimiterea e-mail-ului prin ea. Acest model a funcționat bine zeci de ani, atât timp cât toate calculatoarele din ARPANET (și mai târziu din Internet) erau, de fapt, conectate la rețea și gata să accepte conexiuni TCP.

Totuși, odată cu apariția celor care accesează Internet-ul folosind un modem cu care se conectează la ISP-ul lor, acest lucru nu mai ține. Problema este următoarea: Ce se întâmplă când Elinor vrea să-i trimită Carolynei un e-mail și Carolyn nu este conectată la rețea în acel moment? Elinor nu va putea să stabilească o conexiune TCP cu Carolyn și astfel, nu va putea utiliza protocolul SMTP.

O soluție este ca agentul de transfer de mesaje de pe o mașină ISP să accepte e-mail-ul pentru clienții săi și să-l stocheze în cutiile lor poștale pe o mașină a ISP-ului. Din moment ce acest agent poate fi conectat la rețea tot timpul, se poate trimite e-mail 24 de ore pe zi.

POP3

Din nefericire, această soluție dă naștere altei probleme: cum își ia utilizatorul e-mail-ul de la agentul de transfer de mesaje al ISP-ului? Soluția acestei probleme este crearea unui alt protocol care să permită agenților de transfer mesaje (aflați pe calculatoarele clienților) să contacteze agentul de transfer mesaje (de pe o mașină ISP) și să facă posibilă copierea e-mail-ului de la ISP la utilizator. Un astfel de protocol este **POP3 (Post Office Protocol Version 3**- Protocol de poștă, versiunea 3), definit în RFC 1939.

Situația anterioară (când atât expeditorul cât și destinatarul aveau conexiune permanentă la Internet) este ilustrată în fig. 7-15(a). O situație în care expeditorul este efectiv conectat la rețea (online) dar destinatarul nu, este ilustrată în fig. 7-15(b).

POP3 începe când utilizatorul pornește programul cititor de poștă (mail reader). Acesta sună la ISP (în cază că nu există deja o conexiune) și stabilește o conexiune TCP cu agentul de transfer de mesaje, prin portul 110. Odată ce conexiunea a fost stabilită, protocolul POP3 trece succesiv prin următoarele trei stări:

1. Autorizare.

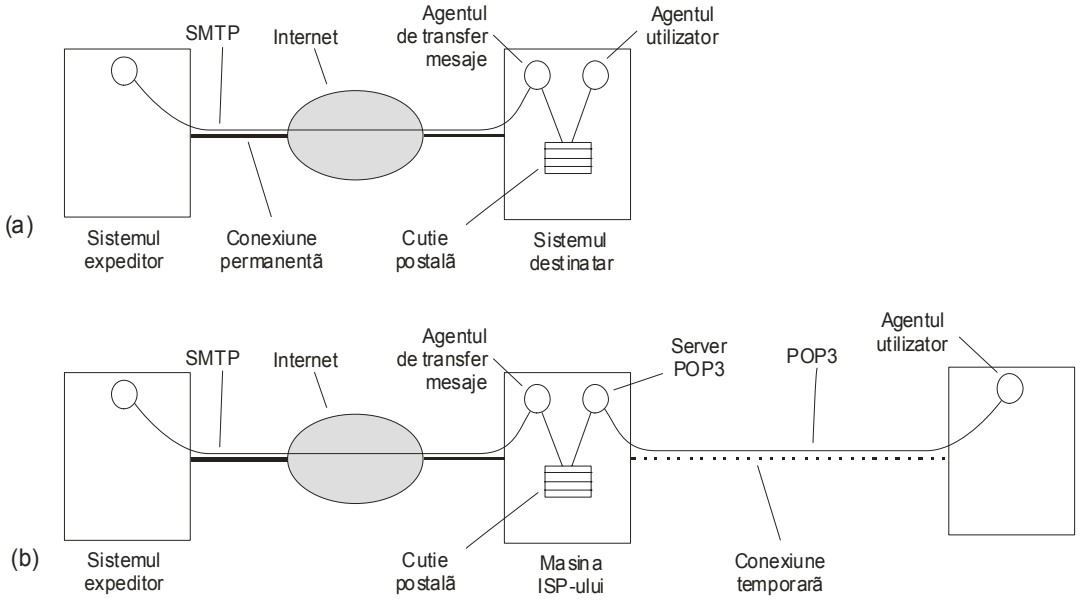


Fig. 7-15. (a) Trimiterea și citirea poștei când destinatarul are o conexiune permanentă la Internet, iar agentul utilizator rulează pe aceeași mașină ca și agentul de transfer de mesaje. (b) Citirea e-mail-ului când destinatarul are o conexiune comutată(dial-up) la un ISP.

2. Tranzacționare.
3. Actualizare.

Starea de autorizare se referă la admiterea utilizatorului în sistem (login). Starea de tranzacționare tratează colectarea e-mail-urilor și marcarea lor pentru ștergere din cutia poștală. Starea de actualizare se ocupă cu ștergerea efectivă a mesajelor.

Această comportare poate fi observată tastând ceva de genul:

```
telnet mail.isp.com 110
```

unde *mail.isp.com* reprezintă numele DNS al serverului de mail de la ISP. Telnet stabilește o conexiune TCP prin portul 110, pe care ascultă serverul POP3. După acceptarea conexiunii TCP, serverul trimite un mesaj ASCII anunțându-și prezența. De obicei, el începe cu +OK urmat de un comentariu. Un exemplu de scenariu este arătat în fig. 7-16 începând după ce conexiunea TCP a fost stabilită. Ca și mai înainte, liniile marcate cu *C:* sunt ale clientului (utilizatorului) iar cele cu *S:* sunt ale serverului (agentul de transfer de mesaje de la ISP).

În timpul stării de autorizare, clientul trimite numele său de utilizator și parola. După conectarea cu succes, clientul poate să trimită comanda *LIST*, care determină serverul să listeze conținutul cutiei poștale. Lista este terminată cu un punct.

Apoi, clientul poate regăsi mesajele folosind comanda *RETR* și le poate marca pentru ștergere cu *DELE*. Când toate mesajele au fost primite (și eventual marcate pentru ștergere), clientul trimite comanda *QUIT* pentru terminarea stării de tranzacționare și intrarea în starea de actualizare. Când serverul a șters toate mesajele, el trimite un răspuns și desființează conexiunea TCP.

```
                    S: +OK Serverul POP3 este pregătit
C: USER carolyn
                    S: +OK
C: PASS vegetables
                    S: +OK autentificare cu succes
C: LIST
                    S: 1 2505
                    S: 2 14302
                    S: 3 8122
                    S: .
C: RETR 1
                    S: (trimite mesajul 1)
C: DELE 1
C: RETR 2
                    S: (trimite mesajul 2)
C: DELE 2
C: RETR 3
                    S: (trimite mesajul 3)
C: DELE 3
C: QUIT
                    S: +OK Serverul POP3 întrerupe legătura
```

Fig. 7-16. Folosirea protocolului POP3 pentru a aduce trei mesaje.

Deși este adevărat că protocolul POP3 are abilitatea de a descărca un anumit mesaj sau un anumit grup de mesaje păstrându-le pe server, cele mai multe programe de e-mail descarcă tot și golesc cutia poștală. Ca urmare, practic singura copie rămâne înregistrată pe discul utilizatorului. Dacă acesta se strică, toate e-mail-urile pot fi pierdute definitiv.

Să recapitulăm pe scurt cum lucrează e-mail-ul pentru clienții unui ISP. Elinor creează un mesaj pentru Carolyn folosind un program de e-mail (adică, agentul utilizator) și face clic pe o icoană pentru a-l trimite. Programul de e-mail trimite mesajul agentului de transfer de mesaje de pe calculatorul Elinorei. Agentul de transfer de mesaje vede că mail-ul este pentru *carolyn@xyz.com* și folosește DNS pentru a căuta înregistrarea MX pentru *xyz.com* (unde *xyz.com* este ISP-ul Carolynei). Această cerere întoarce numele DNS al serverului de mail *xyz.com*. Agentul de transfer de mesaje caută acum adresa IP a acestei mașini folosind din nou DNS, de exemplu *gethostbyname*. Apoi, el stabilește o conexiune TCP cu serverul SMTP pe portul 25 de pe această mașină. Folosind o secvență de comenzi SMTP, asemănătoare celei din fig. 7-14, el transferă mesajul în cutia poștală a Carolynei și întrerupe conexiunea TCP.

După un timp, Carolyn își pornește PC-ul său, se conectează la ISP și pornește programul de e-mail. Programul de e-mail stabilește o conexiune TCP cu serverul POP3 pe portul 110 al serverului de poștă al ISP-ului. Numele DNS sau adresa IP a acestei mașini este configurată în mod normal atunci când programul de e-mail este instalat sau când este făcut contractul cu ISP-ul. După ce conexiunea TCP a fost stabilită, programul de e-mail al Carolynei lansează protocolul POP3 pentru a aduce conținutul cutiei sale poștale pe discul fix folosind comenzi ca cele din fig. 7-16. Odată ce a fost transferat tot e-mail-ul, conexiunea TCP este eliberată. De fapt, conexiunea cu ISP-ul poate fi desființată acum, din moment ce tot e-mailul este pe discul fix al Carolynei. Desigur, pentru a trimite un răspuns, va fi nevoie din nou de conexiunea cu ISP-ul, care, în general, nu este întreruptă imediat după aducerea poștei.

IMAP

Pentru un utilizator cu un singur cont de e-mail, la un singur ISP, care este tot timpul accesat de la un singur PC, POP3 este bun și larg folosit datorită simplității și robusteții sale. Totuși, există în industria calculatoarelor un adevăr bine înrădăcinat, acela că imediat ce un lucru funcționează bine, cineva va începe să ceară mai multe facilități (și să aibă mai multe probleme). Asta s-a întâmplat și cu e-mail-ul. De exemplu, multă lume are un singur cont de e-mail la serviciu sau la școală și vrea să-l acceseze de pe PC-ul de acasă, de pe calculatorul portabil în călătoriile de afaceri și din Internet café-uri în vacanțe. Cu toate că POP3 permite asta, din moment ce în mod normal el descarcă toate mesajele la fiecare conectare, rezultatul constă în răspândirea e-mail-ului utilizatorului pe mai multe mașini, mai mult sau mai puțin la întâmplare, unele dintre ele nefiind ale utilizatorului.

Acest dezavantaj a dat naștere unei alternative a protocolului de livrare finală, **IMAP (Internet Message Access Protocol –** Protocol pentru accesul mesajelor în Internet), care este definit în RFC 2060. Spre deosebire de POP3, care în mod normal presupune că utilizatorul își va goli căsuța poștală la fiecare conectare și va lucra deconectat de la rețea (off-line) după aceea, IMAP presupune că tot e-mail-ul va rămâne pe server oricât de mult, în mai multe căsuțe poștale. IMAP prevede mecanisme extinse pentru citirea mesajelor sau chiar a părților de mesaje, o facilitate folositoare când se utilizează un modem încet pentru citirea părții textuale a unui mesaj cu mai multe părți audio și video de mari dimensiuni. Întrucât premisa de folosire este că mesajele nu vor fi transferate pe calculatorul utilizatorului în vederea stocării permanente, IMAP asigură mecanisme pentru crearea, distrugerea și manipularea mai multor cutii poștale pe server. Astfel, un utilizator poate păstra o cutie poștală pentru fiecare corespondent și poate muta aici mesajele din inbox după ce acestea au fost citite.

IMAP are multe facilități, ca de exemplu posibilitatea de a se referi la un mesaj nu prin numărul de sosire, ca în fig. 7-8, ci utilizând atribute (de exemplu, dă-mi primul mesaj de la Bobbie). Spre deosebire de POP3, IMAP poate de asemenea să accepte atât expedierea mesajelor spre destinație cât și livrarea mesajelor venite.

Stilul general al protocolului IMAP este similar cu cel al POP3-ului, după cum se arată în fig. 7-16, cu excepția faptului că există zeci de comenzi. Serverul IMAP ascultă pe portul 143. În fig. 7-17 este prezentată o comparație între POP3 și IMAP. E bine de notat, totuși, că nu toate ISP-urile oferă ambele protocoale și că nu toate programele de e-mail le suportă pe amândouă. Așadar, atunci când alegeți un program de e-mail, este important să aflați ce protocoale suportă și să vă asigurați că ISP-ul oferă cel puțin unul din ele.

Caracteristica	POP3	IMAP
Unde este definit protocolul	RFC 1939	RFC 2060
Portul TCP folosit	110	143
Unde este stocat e-mail-ul	PC-ul utilizatorului	Server
Unde este citit e-mail-ul	Off-line	On-line
Timpul necesar conectării	Mic	Mare
Folosirea resurselor serverului	Minimă	Intensă
Mai multe cutii poștale	Nu	Da
Cine face copii de siguranță la cutiile poștale	Utilizatorul	ISP-ul
Bun pentru utilizatorii mobili	Nu	Da
Controlul utilizatorului asupra scrisorilor preluate	Mic	Mare
Descărcare parțială a mesajelor	Nu	Da
Volumul discului alocat (disk quota) este o problemă	Nu	Ar putea fi în timp
Simplu de implementat	Da	Nu
Suport răspândit	Da	În creștere

Fig. 7-17. O comparație între POP3 și IMAP.

Facilități de livrare

Indiferent dacă este folosit POP3 sau IMAP, multe sisteme oferă legături pentru procesarea adițională a mesajelor e-mail sosite. Un instrument deosebit de valoros pentru mulți utilizatori de e-mail este reprezentat de capacitatea de a construi filtre. Acestea sunt reguli care se verifică la sosirea mesajelor sau la pornirea agentului utilizator. Fiecare regulă specifică o condiție și o acțiune. De exemplu, o regulă ar putea spune că orice mesaj venit de la șef trebuie pus în cutia poștală numărul 1, orice mesaj de la un anumit grup de prieteni se duce în cutia poștală numărul 2 și orice alt mesaj conținând anumite cuvinte în Subiect este aruncat fără comentarii.

Unii ISP oferă filtre care clasifică automat mesajele sosite ca fiind importante sau nerelevante (spam) și memorează fiecare mesaj în cutia poștală corespunzătoare. Asemenea filtre funcționează verificând mai întâi dacă sursa este un autor cunoscut de mesaje „spam". Apoi examinează subiectul. Dacă sute de utilizatori au primit un mesaj cu același subiect, probabil că el este nerelevant. Există și alte tehnici folosite pentru detecția mesajelor lipsite de importanță.

O altă caracteristică a livrării, pusă la dispoziție adesea, este posibilitatea de a retrimite (temporar) poșta la o adresă diferită. Această adresă poate fi și un calculator utilizat de un serviciu comercial de comunicații, care va contacta utilizatorul prin radio sau satelit, afișând *Subject:* linie pe pagerul său.

O altă trăsătură comună a livrării finale este abilitatea de a instala un **demon de vacanță**. Acesta este un program care examinează fiecare mesaj sosit și trimite o replică insipidă cum ar fi:

Salut. Sunt în vacanță. Mă întorc pe 24 august. O zi bună.

Asemenea răspunsuri pot să specifice, de asemenea, cum să fie tratate problemele urgente, alte persoane care pot fi contactate pentru probleme specifice etc. Majoritatea demonilor de vacanță păstrează urma celor cărora le-au trimis replici și se abțin de la a trimite unei aceleiași persoane o a doua asemenea replică. Demonii buni verifică și dacă mesajul sosit a fost trimis de la o listă de mail și în acest caz, nu mai răspund deloc. (Cei care trimit mesaje în timpul verii la liste mari de e-mail, probabil că nu doresc să primească sute de replici în care să le fie detaliate planurile de vacanță ale fiecăruia.)

Autorul s-a lovit recent de o formă extremă de prelucrare a livrării când a trimis o scrisoare unei persoane care pretinde că primește 600 de mesaje pe zi. Identitatea sa nu va fi deconspirată aici, ca nu cumva jumătate dintre cititorii acestei cărți să-i trimită și ei scrisori. Să-l numim în continuare John.

John și-a instalat un robot de e-mail care verifică fiecare mesaj sosit, ca să vadă dacă este de la un nou corespondent. Dacă este așa, trimite înapoi o replică standard în care explică faptul că nu mai poate să citească personal toate mesajele. În schimb a produs un document FAQ (Frequently Asked Questions) personal, unde răspunde la multe întrebări care i se pun de obicei. În mod normal, grupurile de știri și nu persoanele au documente FAQ.

Documentul FAQ al lui John dă adresa acestuia, numărul de fax și numerele de telefon și spune cum poate fi contactată firma sa. Arată cum poate fi chemat ca vorbitor și explică cum pot fi obținute lucrările sale și alte documente. Furnizează de asemenea referințe la programele scrise de el, o conferință pe care o organizează, un standard al cărui editor este și așa mai departe. E posibil ca această abordare să fie necesară, dar poate că un FAQ personal reprezintă simbolul final al statutului.

Poșta electronică pe Web (Webmail)

Un subiect care merită menționat este poșta electronică pe Web. Anumite situri de Web, cum ar fi Hotmail sau Yahoo oferă servicii de poștă electronică oricui dorește. Ele funcționează după cum

urmează. Au agenți normali de transfer de mesaje, care așteaptă la portul 25 conexiuni noi de SMTP. Pentru a contacta, să spunem Hotmail, trebuie să obțineți înregistrarea sa DNS *MX*, de exemplu tastând

```
host —a —v hotmail.com
```

pe un sistem UNIX. Să presupunem că serverul de poștă electronică se numește *mx10.hotmail.com*; atunci tastând

```
telnet mx10.hotmail.com 25
```

se poate stabili o conexiune TCP prin care se pot trimite comenzi SMTP în modul obișnuit. Deocamdată, nimic special, cu excepția faptului că aceste servere mari sunt adeseori ocupate, ca atare se poate să dureze ceva mai mult până vă este acceptată o cerere de conexiune TCP.

Partea interesantă este cum se transmite poșta electronică. În principiu, atunci când utilizatorul se duce la pagina de Web a poștei electronice, îi este prezentat un formular în care i se cere un nume de cont și o parolă. Când utilizatorul face clic pe **Sign In**, numele de cont și parola sunt trimise serverului, care le validează. Dacă autentificarea s-a făcut cu succes, serverul găsește cutia poștală a utilizatorului și construiește o listă similară cu cea din fig. 7-8, cu diferența că are formatul unei pagini de Web în HTML. Pagina Web este transmisă apoi programului de navigare pentru a fi afișată. Pe multe din elementele paginii se pot executa clic-uri, astfel că mesajele pot fi citite, șterse, ș.a.m.d.

7.3 WORLD WIDE WEB

Web-ul este un context arhitectural pentru accesul la documente, răspândite pe mii de mașini din Internet, între care există legături. În 10 ani a evoluat de la o aplicație pentru transmiterea de date utile pentru fizica energiilor înalte la o aplicație despre care milioane de oameni cred că este Internetul. Popularitatea sa enormă se datorează faptului că are o interfață grafică plină de culoare, ușor de utilizat de către începători și în același timp oferă o cantitate imensă de informație - de la animale mitologice la tribul Zulu, pe aproape orice subiect posibil.

Web-ul (cunoscut și ca **WWW**) a apărut în 1989 la CERN, Centrul European de Cercetări Nucleare. CERN are câteva acceleratoare utilizate de echipe mari de cercetători din țările europene pentru cercetări în fizica particulelor. Deseori aceste echipe au membri din peste zece țări. Majoritatea experiențelor sunt foarte complicate și necesită ani de pregătire și construire de echipamente. Web-ul a apărut din necesitatea de a permite cercetătorilor răspândiți în lume să colaboreze utilizând colecții de rapoarte, planuri, desene, fotografii și alte tipuri de documente aflate într-o continuă modificare.

Propunerea inițială pentru crearea unei colecții de documente având legături între ele (Web) a fost făcută de fizicianul Tim Berners-Lee, fizician la CERN, în martie 1989. Primul prototip (bazat pe text) era operațional 18 luni mai târziu. În decembrie 1991, s-a făcut o demonstrație publică la conferința Hypertex'91 în San Antonio, Texas.

Aceasta demonstrație și publicitatea aferentă au atras atenția altor cercetători, fapt care l-a determinat pe Marc Andreessen de la University of Ilinois să înceapă să dezvolte primul program de navigare grafic, Mosaic. Acesta a fost lansat în februarie 1993. Mosaic a fost atât de popular încât un an mai târziu Marc Andreessen a plecat pentru a forma o nouă companie, Netscape Communications Corp., care se ocupa cu dezvoltarea de software pentru Web. Când Netscape a devenit o com-

panie publică în 1995, investitorii, care probabil că au crezut că este vorba de un fenomen de tip Microsoft, au plătit 1,5 miliarde de dolari pentru acțiunile companiei. Acest record a fost cu atât mai neașteptat cu cât compania avea un singur produs, opera în deficit și anunțase probabilii investitori că nu se așteaptă la beneficii în viitorul apropiat. În următorii trei ani, Netscape Navigator și produsul Internet Explorer al companiei Microsoft au intrat într-un „război al programelor de navigare", fiecare din produse încercând cu frenezie adăugarea de noi opțiuni (și astfel a mai multor erori) decât celălalt. În 1998, America Online a cumpărat Netscape Communications Corp. pentru suma de 4.2 miliarde $, încheind astfel durata scurtă în care Netscape a fost o companie independentă.

În 1994, CERN și M.I.T. au semnat o înțelegere pentru a forma **Consorțiul World Wide Web** (câteodată abreviat ca **W3C**), o organizație care are ca obiectiv dezvoltarea Web-ului, standardizarea protocoalelor, și încurajarea interoperabilității între situri. Berners-Lee a devenit director. De atunci, sute de universități și companii au intrat în consorțiu. M.I.T. coordonează partea americană a consorțiului în timp ce centrul de cercetări francez, INRIA, coordonează partea europeană. Deși există foarte multe cărți despre Web, cel mai bun loc pentru găsirea unor informații la zi despre el este (în mod natural) chiar Web-ul. Pagina consorțiului are adresa *www.w3.org* . Cititorii interesați vor găsi acolo legături la pagini care acoperă toate documentele și activitățile consorțiului.

7.3.1 Aspecte arhitecturale

Din punctul de vedere al utilizatorului, Web-ul constă dintr-o colecție imensă de documente sau **pagini de Web (Web pages)**, adesea numite prescurtat **pagini**, răspândite în toată lumea. Fiecare pagină poate să conțină legături (indicatori) la alte pagini, aflate oriunde în lume. Utilizatorii pot să aleagă o legătură prin execuția unui clic care îi va aduce la pagina indicată de legătură. Acest proces se poate repeta la nesfârșit. Ideea ca o pagină să conțină legături către altele a fost inventată în 1945, cu mult înainte de a se fi inventat Internet-ul, de către Vannevar Bush, un profesor vizionar de la departamentul de inginerie electrică al M.I.T.

Paginile pot să fie văzute cu ajutorul unui **program de navigare** (**browser**). Internet Explorer și Netscape Navigator sunt cele mai cunoscute programe de navigare. Programul de navigare aduce pagina cerută, interpretează textul și comenzile de formatare conținute în text și afișează pagina, formatată corespunzător, pe ecran. Un exemplu este prezentat în fig. 7-18(a). Ca majoritatea paginilor de Web, începe cu un titlu, conține informații și se termină cu adresa de poștă electronică a celui care menține pagina. Șirurile de caractere care reprezintă legături la alte pagini, se numesc **hiper-legături**, sunt afișate în mod diferit, fiind subliniate și/sau colorate cu o culoare specială. Pentru a selecta o legătură, utilizatorul va plasa cursorul pe zona respectivă, ceea ce va determina schimbarea formei cursorului și va executa un clic. Deși există programe de navigare fără interfață grafică, ca de exemplu Lynx, ele nu sunt atât de utilizate ca programele de navigare grafice, astfel încât în continuare ne vom referi numai la ultimele. Au fost dezvoltate și programe de navigare bazate pe voce.

Utilizatorii care sunt interesați să afle mai multe despre „Department of Animal Psichology" vor selecta numele respectiv (apare subliniat). Programul de navigare va aduce pagina la care este legat numele respectiv și o va afișa, așa cum se vede în fig. 7-18(b). Șirurile subliniate aici pot să fie selectate la rândul lor pentru a aduce alte pagini și așa mai departe. Noua pagină se poate afla pe aceeași mașină ca și prima sau pe o mașină aflată undeva pe glob la polul opus. Utilizatorul nu va ști. Aducerea paginilor este realizată de către programul de navigare, fără nici un ajutor din partea utilizatorului. Dacă utilizatorul se va întoarce la prima pagină, legăturile care au fost deja utilizate vor fi afișate

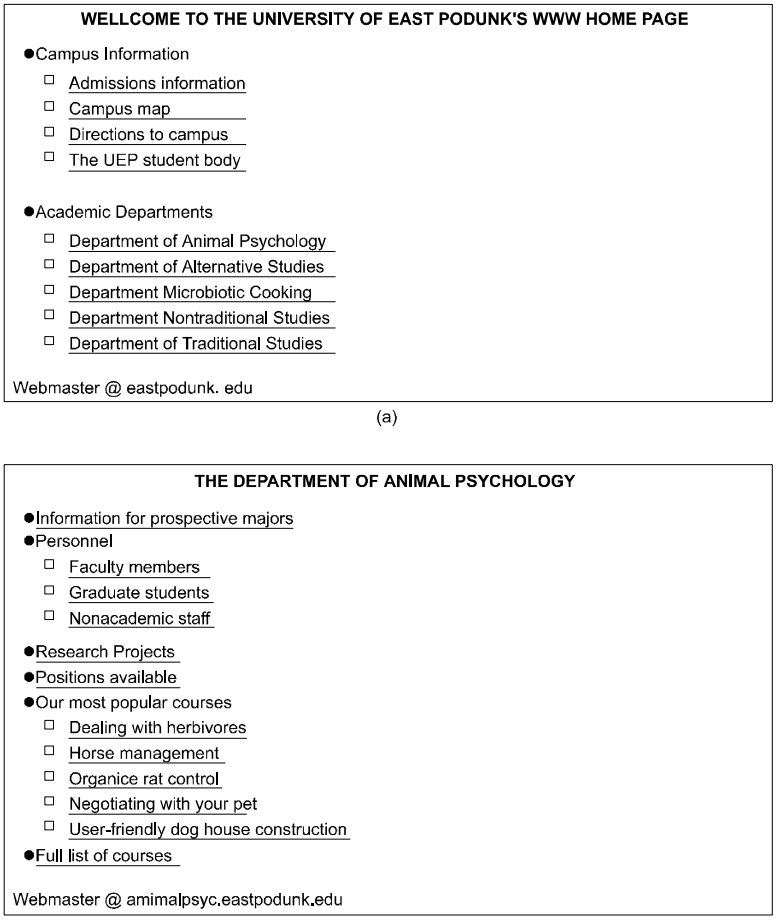


Fig. 7-18. (a) O pagină de Web. (b) pagina la care se ajunge dacă se selectează Department of Animal Psychology

altfel decât celelalte (subliniate cu linie punctată sau utilizând o altă culoare) pentru a fi deosebite de cele care nu au fost încă selectate. De notat că selecția liniei *Campus Information* din prima pagină nu are nici un efect. Nu este subliniată, ceea ce înseamnă că este pur și simplu un text care nu este legat de o altă pagină.

Modelul de bază al funcționării Web-ului este arătat in fig. 7-19. Aici un program de navigare afișează o pagină de Web pe mașina clientului. Atunci când utilizatorul face clic pe linia de text ce indică spre o pagină de pe serverul *abcd.com*, programul de navigare urmează hiper-legătura trimițând un mesaj serverului *abcd.com* în care se cere pagina respectivă. Atunci când pagina sosește, ea este afișată. Dacă această pagină conține o hiper-legătură către o pagină de pe serverul *xyz.com* pe care utilizatorul face clic, programul de navigare trimite o cerere mașinii respective pentru acea pagină, și așa mai departe la nesfârșit.

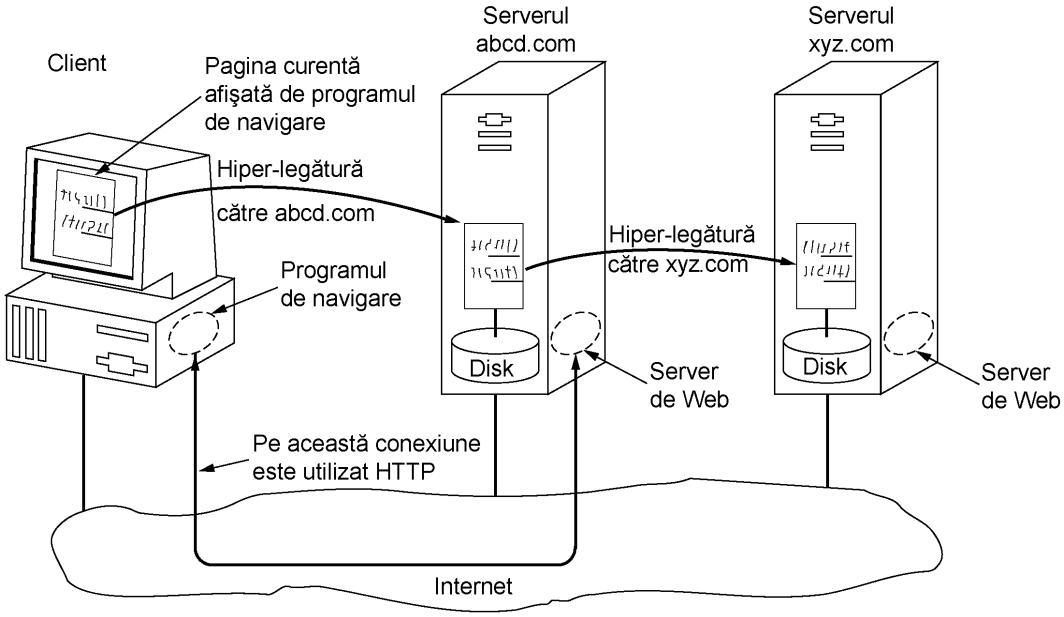


Fig. 7-19. Părţile componente ale modelului Web-ului

Aspecte privind clientul

Să examinăm acum în detaliu aspectele ce privesc clientul din fig. 7-19. În esenţă, un program de navigare este o aplicaţie capabilă să afişeze o pagină de Web şi să capteze clicurile mouse-ului pe elemente ale paginii afişate. Când un element este selectat, programul de navigare urmează hiper-legătura şi obţine pagina selectată. Ca atare, hiper-legătura conţinută în pagină necesită o modalitate de a adresa prin nume orice altă pagină de pe Web. Paginile sunt adresate prin nume folosind **URL-uri (Uniform Resource Locators,** rom.: **Localizatoare Uniforme de Resurse)**. Un URL tipic este

http://www.abcd.com/products.html

Vom explica ce înseamnă URL mai târziu, în cadrul capitolului curent. Deocamdată, este suficient să ştim că un URL are trei părţi: numele protocolului *(http)*, numele calculatorului pe care se găseşte pagina (*www.abcd.com*) şi numele fişierului care conţine pagina (*products.html*).

Când un utilizator execută un clic pe o hiper-legătură, programul de navigare urmează o serie de etape pentru a obţine pagina indicată de hiper-legătură. Să presupunem ca utilizatorul navighează pe Web şi găseşte o legătura despre telefonia pe Internet care indică spre pagina principală a ITU, *http://www.itu.org/home/index.html*. Să urmăm etapele parcurse când această legătură este selectată.

1. Programul de navigare determină URL (pe baza selecţiei).
2. Programul de navigare întreabă DNS care este adresa IP pentru *www.itu.org*.
3. DNS răspunde cu 156.106.192.32.
4. Programul de navigare realizează conexiunea TCP cu portul 80 al 156.106.192.32.
5. Trimite o cerere pentru fişierul */home/index.html.*
6. Serverul *www.itu.org* transmite fişierul */home/index.html.*
7. Conexiunea TCP este eliberată.
8. Programul de navigare afişează textul din */home/index.html.*
9. Programul de navigare aduce şi afişează toate imaginile din acest fişier.

Multe programe de navigare informează despre etapa care se execută într-o fereastră de stare, în partea de jos a paginii. În acest mod, dacă performanțele sunt slabe, utilizatorul poate să știe dacă este vorba de faptul că DNS nu răspunde, că serverul nu răspunde, sau pur și simplu de congestia rețelei în timpul transmisiei paginii.

Pentru a afișa noua pagină (sau orice pagină), programul de navigare trebuie să înțeleagă forma în care este scrisă. Pentru a permite tuturor programelor de navigare să înțeleagă orice pagină de Web, paginile de Web sunt scrise într-un limbaj standardizat numit HTML, care descrie paginile de Web. Vom discuta acest limbaj mai târziu în acest capitol.

Deși un program de navigare este în principiu un interpretor de HTML, majoritatea programelor de navigare au numeroase butoane și opțiuni care ajută navigarea prin Web. Multe au un buton pentru revenirea la pagina anterioară, un buton pentru a merge la pagina următoare (acest buton este operațional numai după ce utilizatorul s-a întors înapoi dintr-o pagină) și un buton pentru selecția paginii personale (home page). Majoritatea programelor de navigare au un buton sau un meniu pentru înregistrarea unei adrese de pagină (bookmark) și un altul care permite afișarea unor adrese înregistrate, făcând astfel posibilă revenirea la o pagină cu ajutorul câtorva selecții simple realizate cu mausul. Paginile pot să fie salvate pe disc sau tipărite. Sunt disponibile numeroase opțiuni pentru controlul ecranului și configurarea programului de navigare conform dorințelor utilizatorului.

În afară de text obișnuit (nesubliniat) și hipertext (subliniat), paginile de Web pot să conțină iconițe, desene, hărți, fotografii. Fiecare dintre acestea poate să fie, în mod opțional, legată la altă pagină. Dacă se selectează unul dintre aceste elemente, programul de navigare va aduce pagina respectivă și o va afișa, așa cum se întâmplă în cazul selectării unui text. Pentru imaginile care sunt fotografii sau hărți, alegerea paginii care se aduce poate să depindă de regiunea din imagine pe care se face selecția.

Nu toate paginile conțin HTML. O pagină poate fi formată dintr-un document în format PDF, o iconiță în format GIF, o fotografie în format JPEG, o melodie în format MP3, o înregistrare video în format MPEG sau oricare din cele alte câteva sute de tipuri de fișiere. Deoarece paginile în forma standard HTML pot avea legături către oricare din acestea, programul de navigare are o problemă atunci când întâlnește o pagină pe care nu o poate interpreta.

În loc sa facă programele de navigare din ce în ce mai mari, înglobând interpretoare pentru o colecție de tipuri de fișiere în creștere rapidă, majoritatea programelor de navigare au ales o soluție mai generală. Atunci când un server întoarce o pagină, el întoarce de asemenea informații adiționale despre acea pagină. Această informație include tipul MIME al paginii (fig. 7-12). Paginile de tipul *text/html* sunt afișate direct, ca și paginile de alte câteva tipuri interpretate implicit. Dacă tipul MIME nu este unul dintre acestea, programul de navigare își consultă tabela de tipuri MIME pentru a afla cum să afișeze pagina. Această tabelă asociază un tip MIME cu o aplicație de vizualizare.

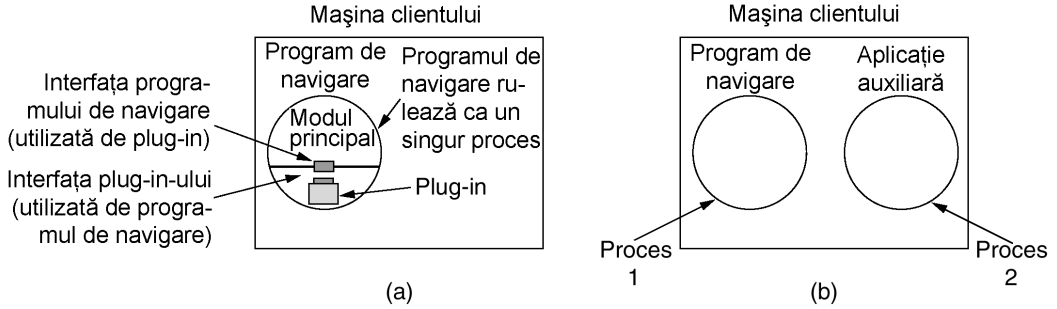


Fig. 7-20. (a) Un plug-in al programului de navigare. (b) O aplicație auxiliară

Există două posibilități: plug-in-uri și programe auxiliare (*helper applications*). Un **plug-in** este un modul pe care programul de navigare îl obține dintr-un director special de pe disc și îl instalează ca o extensie al însuși programului de navigare, așa cum se arată în fig. 7-20(a). Deoarece plug-in-urile se execută în interiorul programului de navigare, acestea au acces la pagina curentă și pot să modifice modul în care aceasta este afișată. După ce plug-in-ul a terminat ceea ce avea de făcut (de obicei după ce utilizatorul s-a deplasat la altă pagină de Web), acesta este scos din memoria programului de navigare.

Fiecare program de navigare are o colecție de proceduri pe care toate plug-in-urile trebuie să le implementeze pentru ca programul de navigare să poată executa plug-in-ul. De exemplu, există de obicei o procedură pe care modulul principal al programului de navigare o apelează pentru a oferi plug-in-ului date ce trebuie afișate. Această colecție de proceduri constituie interfața unui plug-in și este particulară fiecărui program de navigare.

În plus, programul de navigare pune la dispoziția plug-in-ului propria sa colecție de proceduri. Procedurile tipice din interfața programului de navigare se referă la alocarea și eliberarea memoriei, afișarea de mesaje în fereastra de stare a programului de navigare sau obținerea parametrilor acestuia.

Înainte ca un plug-in să poată fi folosit, acesta trebuie instalat. Procedura uzuală de instalare este ca utilizatorul să navigheze la situl de Web al plug-in-ului și să copieze un fișier de instalare. Pe sistemele Windows, acesta este de obicei o arhivă zip cu decompresie automată cu extensia *.exe*. Când pe acest fișier se execută un dublu clic, se execută un program de mici dimensiuni atașat părții de început a fișierului. Acest program decomprimă plug-in-ul și îl copiază în directorul de plug-in-uri al programului de navigare. Apoi execută apelurile necesare pentru înregistrarea tipului MIME corespunzător și asocierea acestuia cu plug-in-ul. Pe sistemele UNIX, fișierul de instalare este in mod frecvent un fișier de comenzi care se ocupă de copiere și înregistrare.

Cea de-a doua modalitate de extindere a unui program de navigare este utilizarea **aplicațiilor auxiliare (helper applications)**. Acestea sunt programe complete ce se execută ca procese separate. Acest fapt este ilustrat în fig. 7-20(b). Deoarece acestea sunt programe separate, nu oferă nici o interfață programului de navigare și nu utilizează serviciile acestuia. De obicei însă acceptă doar numele unui fișier temporar unde a fost stocat conținutul paginii, deschide acest fișier și îi afișează conținutul. De obicei, aplicațiile auxiliare sunt programe de dimensiuni mari care există independent de programul de navigare, cum ar fi Adobe Acrobat Reader pentru afișarea fișierelor PDF, sau Microsoft Word. Unele programe (cum ar fi Acrobat) dispun de un plug-in care execută aplicația auxiliară.

Multe aplicații auxiliare folosesc tipul MIME *application*. A fost definit un număr considerabil de subtipuri, de exemplu *application/pdf* pentru fișiere PDF și *application/msword* pentru fișiere Word. În acest mod, un URL poate să indice direct către un fișier PDF sau Word și atunci când utilizatorul execută un clic asupra sa aplicațiile Acrobat sau Word sunt pornite automat și li se transmite numele fișierului temporar ce conține datele ce trebuie afișate. Ca atare, programele de navigare pot fi configurate să trateze un număr teoretic nelimitat de tipuri de documente, fără schimbări aduse programului de navigare. Serverele de Web moderne sunt adesea configurate cu sute de combinații de tipuri/subtipuri și combinații noi sunt adăugate de fiecare dată când este instalat un program nou.

Aplicațiile auxiliare nu sunt restricționate la utilizarea tipului MIME *application*. Adobe Photoshop folosește *image/x-photoshop* și RealOne Player poate trata de exemplu *audio/mp3*.

Pe sistemele Windows, atunci când un program este instalat pe un calculator, el înregistrează tipurile MIME pe care dorește să le trateze. Acest mecanism conduce la conflicte atunci când mai multe aplicații sunt disponibile pentru vizualizarea unui subtip, cum ar fi *video/mpg*. Ceea ce se în-

tâmplă este că ultimul program ce se înregistrează suprainscrie asociațiile existente (tip MIME, aplicație auxiliară), captând tipul pentru sine. Ca o consecință, instalarea unui nou program poate schimba modul în care un program de navigare tratează tipurile existente.

Pe sistemele UNIX, acest proces de înregistrare nu se face în general automat. Utilizatorul trebuie să schimbe anumite fișiere de configurare. Această abordare conduce la un volum mai mare de muncă dar la surprize mai puține.

Programele de navigare pot de asemenea deschide fișiere locale în loc de a le obține de pe servere de Web de la distanță. Deoarece fișierele locale nu au tipuri MIME, programul de navigare necesită o metodă pentru a determina ce plug-in sau aplicație auxiliară trebuie folosit pentru alte tipuri decât cele tratate implicit ca *text/html* sau *image/jpeg*. Pentru tratarea fișierelor locale, aplicațiile auxiliare pot fi asociate și cu o extensie de fișier, ca și cu un tip MIME. Considerând configurația standard, deschiderea fișierului *foo.pdf* îl va încărca în Acrobat și deschiderea fișierului *bar.doc* îl va încărca în Word. Anumite programe de navigare folosesc tipul MIME, extensia fișierului și chiar informații din interiorul fișierului pentru a ghici tipul MIME. În special Internet Explorer se bazează în primul rând pe extensia fișierului decât pe tipul MIME atunci când acest lucru este posibil.

Și aici pot apare conflicte deoarece multe programe sunt dispuse, de fapt dornice să trateze *mpg*, de exemplu. În timpul instalării, programele create pentru profesioniști afișează frecvent căsuțe de selecție pentru tipurile MIME și extensiile pe care sunt pregătite să le trateze pentru a permite utilizatorului selecția celor dorite pentru a preveni astfel suprainscrierea accidentală a asociațiilor existente. Programele ce au ca țintă marea masă a consumatorilor presupun că utilizatorul nu știe ce este un tip MIME și pur și simplu acaparează tot ce pot fără să țină seama de ceea ce au făcut programele instalate anterior.

Capacitatea de a extinde programul de navigare cu un număr mare de tipuri noi este utilă, dar poate duce și la probleme. Atunci când Internet Explorer obține un fișier cu extensia *exe* își da seama că acest fișier este un program executabil și ca atare nu are o aplicație adițională asociată. Acțiunea evidentă este execuția fișierului. Însa această acțiune poate fi o problemă de securitate enormă. Tot ceea ce trebuie să facă un site de Web rău intenționat este să ofere o pagină de Web, de exemplu cu fotografii de staruri de cinema sau sportive, toate imaginile indicând către un virus. Un singur clic pe o imagine determină ca un program executabil necunoscut și posibil ostil să fie copiat și executat pe mașina utilizatorului. Pentru a preveni astfel de vizitatori nedoriți, Internet Explorer poate fi configurat să fie selectiv în a executa programe necunoscute în mod automat, dar nu toți utilizatorii înțeleg cum să folosească această configurare.

Pe sistemele UNIX pot exista probleme similare cu fișierele de comenzi pentru consola sistemului, dar aceasta necesită ca utilizatorul să instaleze în mod conștient consola sistemului ca aplicație auxiliară. Din fericire, acest proces este suficient de complicat pentru ca nimeni să nu poată să îl efectueze accidental (și puține persoane pot să îl efectueze chiar intenționat).

Aspecte privind serverul

Cam atât despre aspectele privind clientul. Să ne referim acum la aspectele privind serverul. Așa cum am văzut mai sus, atunci când utilizatorul tastează un URL sau execută un clic asupra unei linii de hipertext, programul de navigare analizează URL-ul și interpretează partea între *http://* și următorul caracter / ca un nume DNS ce trebuie căutat. Înarmat cu adresa IP a serverului, programul de navigare stabilește o conexiune TCP la portul 80 de pe acel server. Apoi se transmite o comandă ce conține restul URL-ului, care este de fapt numele fișierului de pe acel server. Serverul întoarce apoi fișierul pentru a fi afișat de către programul de navigare.

Într-o primă aproximare, un server de Web este similar cu serverul din fig. 6-6. Acel server, ca și un server de Web real, primește numele fișierului ce trebuie căutat și transmis programului de navigare. În ambele cazuri, etapele pe care le parcurge serverul în bucla sa principală sunt:

1. Acceptă o conexiune TCP de la un client (program de navigare).
2. Obține numele fișierului cerut.
3. Obține fișierul (de pe disc).
4. Întoarce fișierul clientului.
5. Eliberează conexiunea TCP.

Serverele de Web moderne au mai multe caracteristici, dar în esență acestea sunt funcțiile unui server de Web. O problemă cu această arhitectură este că fiecare cerere necesită acces la disc pentru obținerea fișierului. Rezultatul este că serverul de Web nu poate servi mai multe cereri pe secundă decât numărul de accese la disc ce se pot executa pe secundă. Un disc SCSI are un timp de acces mediu de circa 5 ms, ceea ce limitează serverul la cel mult 200 de cereri/sec, chiar mai puțin dacă trebuie citite des fișiere mari. Pentru un sit de Web de importanță mare, acest număr este prea mic.

O îmbunătățire evidentă (utilizată de toate serverele de Web) este folosirea unui sistem de memorie ascunsă, temporară pentru cele mai recente *n* fișiere utilizate. Înainte de obținerea unui fișier de pe disc, serverul verifică memoria ascunsă (cache). Dacă fișierul există acolo, el poate fi servit direct din memorie, eliminând astfel accesul la disc. Deși pentru o memorie ascunsă eficientă sunt necesare o cantitate mare de memorie principală și timp de procesare suplimentară pentru a analiza memoria ascunsă și pentru a-i administra conținutul, economia de timp este aproape întotdeauna superioară timpului suplimentar de procesare și costului memoriei.

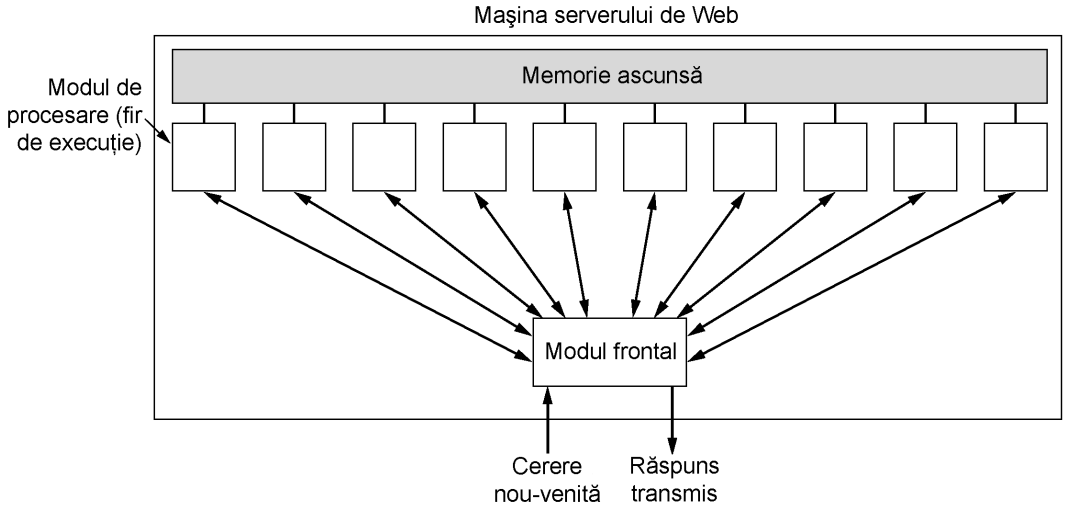


Fig. 7-21. Un server de Web cu mai multe fire de execuție cu un modul frontal și module de procesare

Următorul pas pentru construcția unui server mai rapid este de a face serverul să admită mai multe fire de execuție (*multithreaded*). Într-una din arhitecturi, serverul este format dintr-un modul frontal (*front-end module*), care acceptă conexiunile nou venite, și *k* module de procesare, așa cum arata fig. 7-21. Cele $k + 1$ fire de execuție aparțin toate aceluiași proces, astfel că modulele de proce-

sare au toate acces la memoria ascunsă din interiorul spațiului de adrese al procesului. La sosirea unei cereri, modulul frontal o acceptă și construiește o scurtă înregistrare ce descrie cererea. Aceasta este transmisă apoi unuia dintre modulele de procesare. În altă arhitectură posibilă, modulul frontal este eliminat și fiecare modul de procesare încearcă să își obțină propriile cereri, dar în acest caz este necesar un protocol de sincronizare pentru prevenirea conflictelor.

Modulul de procesare verifică mai întâi memoria ascunsă pentru a determina dacă fișierul necesar se află acolo. Dacă da, modifică înregistrarea pentru a include și un indicator către fișierul din înregistrare. Dacă fișierul nu se află acolo, modulul de procesare începe operațiile cu discul pentru a citi fișierul în memoria ascunsă (renunțând eventual la alte fișiere pentru a face loc acestuia). Când fișierul este citit de pe disc, el este pus în cache și de asemenea transmis clientului.

Avantajul acestei scheme este că în timp ce unul sau mai multe module de procesare sunt blocate așteptând terminarea operațiilor cu discul (și deci nu consumă din timpul procesorului), alte module pot fi active lucrând la satisfacerea altor cereri. Desigur, pentru a obține o îmbunătățire reală asupra modelului cu un singur fir de execuție este necesară existența mai multor unități de disc, astfel încât mai multe discuri să poată fi ocupate în același timp. Cu ajutorul a k module de procesare și k unități de disc, eficiența poate crește până la de k ori față de modelul serverului cu un singur fir de execuție și o singură unitate de disc.

Teoretic, un server cu un singur fir de execuție și k unități de disc poate de asemenea câștiga un factor k în ceea ce privește eficiența, dar implementarea și administrarea sunt mult mai complicate deoarece apelurile de sistem READ normale, blocante nu pot fi folosite pentru accesul la disc. În cazul unui server cu mai multe fire de execuție, acestea pot fi folosite deoarece o operație READ blochează doar firul de execuție care a executat operația și nu întregul proces.

Serverele de Web moderne efectuează mai multe operații decât acceptarea numelor de fișiere și transmiterea conținutului acestora. De fapt, procesarea fiecărei cereri poate deveni destul de complicată. Din acest motiv, într-un număr mare de servere fiecare modul de procesare efectuează o serie de etape. Modulul frontal transmite fiecare cerere sosită către primul modul de procesare disponibil, care apoi execută cererea, utilizând o submulțime a următorilor pași, în funcție de ce pași sunt necesari pentru respectiva cerere.

1. Rezolvarea numelui paginii de Web cerute.
2. Autentificarea clientului.
3. Verificarea drepturilor de acces ale clientului.
4. Verificarea drepturilor de acces asupra paginii de Web.
5. Verificarea memoriei ascunse.
6. Obținerea paginii cerute, de pe disc.
7. Determinarea tipului MIME ce va fi inclus în răspuns.
8. Rezolvarea altor probleme minore.
9. Transmiterea răspunsului către client.
10. Adăugarea unei înregistrări în jurnalul serverului.

Pasul 1 este necesar deoarece cererea sosită poate să nu conțină numele propriu-zis al fișierului, ca șir de caractere. De exemplu, putem considera URL-ul *http://www.cs.vu.nl*, care are un nume de fișier vid. Acesta trebuie extins la un nume de fișier implicit. De asemenea, programele de navigare moderne pot specifica limba implicită a utilizatorului (de ex.: italiană sau engleză), ceea ce deschide posibilitatea ca serverul să selecteze o pagină de Web în acea limbă, dacă aceasta este disponibilă. În

general, extinderea numelor nu este un proces atât de banal cum ar putea părea la prima vedere, datorită unei varietăți de convenții existente privind numirea fișierelor.

Pasul 2 constă în verificarea identității clientului. Acest pas este necesar pentru paginile care nu sunt disponibile publicului larg. Vom discuta o modalitate de a realiza acest lucru mai târziu, în acest capitol.

Pasul 3 verifică dacă există restricții referitoare la satisfacerea cererii, având în vedere identitatea și localizarea clientului. Pasul 4 verifică dacă există restricții de acces asociate cu pagina însăși. Dacă un anumit fișier (de ex.: *.htaccess*) este prezent în directorul unde se află și pagina dorită, accesul la acel fișier poate fi restrâns la anumite domenii, de exemplu numai la utilizatorii din interiorul companiei.

Pașii 5 și 6 presupun obținerea paginii. Pasul 6 necesită capacitatea de tratare simultană a mai multor citiri de pe disc.

Pasul 7 se referă la determinarea tipului MIME din extensia fișierului, primele câteva cuvinte din fișier, un fișier de configurare sau alte surse posibile. Pasul 8 este destinat unei diversități de operații, cum ar fi construcția unui profil al utilizatorului sau adunarea unor statistici.

Pasul 9 este cel în care rezultatul este transmis clientului și pasul 10 adaugă o înregistrare în jurnalul sistemului, în scopuri administrative. Asemenea fișiere de jurnalizare pot fi analizate ulterior pentru obținerea de informații importante despre comportamentul utilizatorului, spre exemplu ordinea în care vizitatorii accesează paginile.

Dacă sosesc prea multe cereri în fiecare secundă, procesorul nu va fi capabil sa suporte încărcarea, oricâte unități de disc ar fi utilizate în paralel. Soluția este adăugarea mai multor noduri (calculatoare), posibil cu unități de disc replicate pentru a evita ca discurile să devină următorul punct de gâtuire. Acest fapt conduce la modelul **fermei de servere (server farm)** din fig. 7-22. Un modul frontal acceptă în continuare cererile dar le împarte mai multor procesoare, nu mai multor fire de execuție, pentru a reduce încărcarea pe fiecare calculator. Mașinile individuale pot fi cu mai multe fire de execuție și în bandă de asamblare ca mai sus.

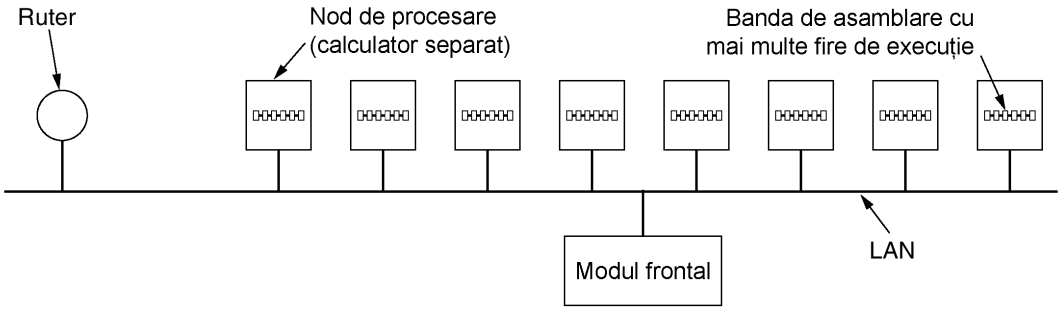


Fig. 7-22. O fermă de servere

O problemă cu fermele de servere este că nu mai există o memorie ascunsă partajată deoarece fiecare nod are propria sa memorie – decât dacă se folosește un sistem multiprocesor cu memorie partajată de cost mic. O modalitate de a contracara această pierdere de performanță este un modul frontal care reține unde direcționează fiecare cerere și trimite cererile ulterioare pentru aceeași pagină aceluiași nod. Această abordare specializează fiecare nod în tratarea anumitor pagini astfel încât spațiul destinat pentru cache nu se pierde reținând fiecare fișier în fiecare cache.

O altă problemă cu fermele de servere este aceea că fiecare conexiune TCP a clientului se termină la modulul de intrare, astfel că răspunsul trebuie transmis prin acest modul. Această situație este

evidențiată în fig. 7-23(a), unde cererea sosită (1) și răspunsul transmis (4) trec ambele prin modulul frontal. Câteodată se poate folosi o soluție ingenioasă numită **parsare TCP (eng.: TCP handoff)** pentru a evita această problemă. Cu acest truc, capătul comunicației TCP este „pasat" nodului de procesare astfel că acesta poate replica direct clientului, lucru evidențiat ca (3) în fig. 7-23(b). Această pasare este făcută într-un mod transparent pentru client.

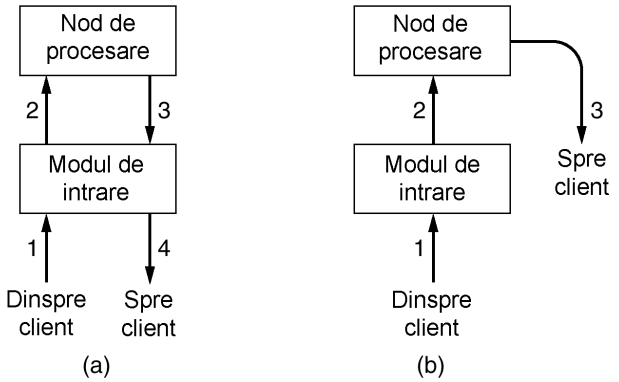


Fig. 7-23. (a) Secvență normală de mesaje cerere – răspuns. (b) Secvență în care se folosește pasarea TCP.

URL- Uniform Resource Locators

Am spus de mai multe ori că o pagină de Web poate să conțină referințe la alte pagini. Să explicăm cum sunt implementate aceste referințe. Încă de la crearea Web-ului, a fost clar că pentru a avea o pagină care să indice spre altă pagină este necesar un mecanism care să permită numirea și regăsirea paginilor. În particular, sunt trei întrebări la care trebuie să se răspundă înainte de a se putea afișa o pagină:

1. Cum se numește pagina?
2. Cum este localizată pagina?
3. Cum se face accesul la pagină?

Dacă fiecare pagină ar avea un nume unic, atunci nu ar exista nici o ambiguitate în identificarea paginilor. Totuși, problema nu este încă rezolvată. Să considerăm de exemplu o paralelă între oameni și pagini. În SUA aproape fiecare persoană are un număr de asigurare socială, care este un identificator unic, astfel încât nu există două persoane cu același număr. Totuși, cunoscând numai numărul respectiv, nu există nici o posibilitate de a găsi adresa persoanei respective și sigur nu se poate afla dacă persoanei respective trebuie să i se scrie în Engleză, Spaniolă sau Chineză. Web-ul are practic același fel de probleme.

Soluția aleasă identifică paginile într-un mod care rezolvă toate trei problemele în același timp. Fiecare pagină are un **URL (Uniform Resource Locator -** adresa uniformă pentru localizarea resurselor) care funcționează ca nume al paginii general valabil. Un URL are trei componente: protocolul (cunoscut și sub numele de **schemă**), numele DNS al mașinii pe care este memorat fișierul și un nume local, care indică în mod unic pagina (de obicei numele fișierului care conține pagina). De exemplu, situl de Web al departamentului din care face parte autorul conține un număr de înregistrări video despre universitate si despre orașul Amsterdam. URL-ul paginii cu înregistrările video este:

http://www.cs.vu.nl/video/index-en.html

Acest URL este format din trei componente: protocolul (*http*), numele DNS al serverului (*www.cs.vu.nl*) și numele fișierului (*video/index-en.html*), cu semnele de punctuație corespunzătoare. Numele fișierului este o cale relativă la directorul de Web implicit de la *cs.vu.nl*.

Se utilizează notații care reprezintă prescurtări standard. În cazul multor situri, un nume de fișier nul înseamnă implicit pagina principală a organizației. În mod obișnuit, atunci când numele fișierului denotă un director, aceasta implică un fișier numit *index.html.* În sfârșit, *~user/* poate să fie pus în corespondență cu directorul WWW al utilizatorului *user*, și apoi cu fișierul *index.html* în acest director. De exemplu, pagina autorului poate să fie referită ca:

http://www.cs.vu.nl/~ast/

chiar dacă de fapt numele propriu-zis al fișierului este *index.html,* implicit în acest director.

Acum ar trebui să fie clar cum funcționează hipertextul. Pentru a face o porțiune de text selectabilă, cel care scrie pagina trebuie să furnizeze două elemente: textul prin care se face selecția și URL-ul paginii care trebuie adusă, dacă textul este selectat. Vom explica sintaxa comenzii mai târziu în acest capitol.

Când se face selecția, programul de navigare caută numele serverului utilizând DNS-ul. Pe baza adresei IP a serverului, programul de navigare stabilește o conexiune TCP spre server. Utilizând această conexiune, se transmite numele fișierului utilizând protocolul specificat. Bingo. Acum sosește pagina.

Această schemă URL este deschisă în sensul că este simplu să se utilizeze alte protocoale pentru a se obține diferite tipuri de resurse. De fapt au fost definite URL-uri pentru protocoalele obișnuite, și multe programe de navigare înțeleg aceste protocoale. Forme simplificate ale celor mai obișnuite sunt prezentate în fig. 7-24.

Nume	Utilizat pentru	Exemple
http	Hipertext (HTML)	http://www.cs.vu.nl/~ast
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
File	Fișier local	file:///usr/suzanne/prog.c
news	Grup de știri	news:AA0134223112@cs.utah.edu
news	Articol de știri	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/libraries
mailto	Trimitere de poșta electronică	mailto:JohnUser@acm.org
telnet	Conectare la distanță	telnet://www.w3.org:80

Fig. 7-24. Câteva URL-uri obișnuite.

Să parcurgem lista rapid. Protocolul *http* este protocolul nativ pentru Web, el este utilizat de către serverele de Web. **HTTP** este o prescurtare pentru **HyperText Transfer Protocol**. Vom examina mai detaliat acest protocol mai târziu în acest capitol.

Protocolul *ftp* este utilizat pentru accesul la fișiere prin FTP (File Transfer Protocol - protocol pentru transferul de fișiere), protocolul Internet de transfer de fișiere. FTP este utilizat de peste douăzeci de ani și este foarte răspândit. Numeroase servere de FTP din toată lumea permit ca de oriunde din Internet să se facă o conectare și să se aducă orice fișier plasat pe un server FTP. Web-ul nu aduce schimbări aici, face doar ca obținerea fișierelor să se facă mai ușor, pentru că FTP are o interfață mai puțin prietenoasă (dar este mai puternic decât HTTP, deoarece permite de exemplu ca un utilizator de pe mașina A să transfere un fișier de pe mașina B pe mașina C).

Este posibil să se facă acces la un fișier local ca la o pagină de Web, fie utilizând protocolul *file* (fișier*)*, fie pur și simplu utilizând numele fișierului. Această abordare este similară utilizării protocolului FTP, dar nu implică existența unui server. Desigur funcționează numai pentru fișiere locale, nu și pentru cele aflate la distanță.

Cu mult înainte de apariția Internet-ului exista sistemul de știri USENET. Acesta este format din aproximativ 30000 de grupuri de știri în care milioane de persoane discută despre o mare varietate de subiecte, adăugând și citind articole legate de subiectul grupului de știri. Protocolul *news* permite citirea un articol din știri ca și cum ar fi o pagină de Web. Aceasta înseamnă că un program de navigare este în același timp și un cititor de știri. De fapt, multe programe de navigare au butoane sau elemente de meniu care permit citirea știrilor USENET mai ușor decât dacă se utilizează cititoare standard de știri.

Protocolul *news* admite două formate. Primul format specifică un grup de știri și poate să fie utilizat pentru a obține o listă de articole de la un server de știri preconfigurat. Al doilea format cere identificatorul unui articol, de exemplu *AA0134223112@cs.utah.edu*. Programul de navigare aduce articolul de la serverul corespunzător utilizând protocolul **NNTP (Network News Transfer Protocol – Protocol de transfer al știrilor prin rețea)**. Nu vom studia NNTP în această carte, dar este în mare bazat pe SMTP și are un stil similar.

Protocolul *gopher* era utilizat de sistemul Gopher, care a fost proiectat la universitatea Minnesota. Numele este cel al echipei atletice a universității, the Golden Gopher (de asemenea acest nume este utilizat în argou pentru „go for" adică o comandă de aducere). Gopher-ul a precedat Web-ul cu câțiva ani. Era o metodă de regăsire a informației, similară conceptual cu cea utilizată de Web, dar acceptând numai text și imagini. Este considerat depășit și nu se mai folosește în prezent.

Ultimele două protocoale nu sunt de fapt protocoale pentru aducerea unor pagini de Web, dar sunt utile. Protocolul *mailto* permite transmiterea de poștă dintr-un program de navigare. Pentru a face această operație, se selectează butonul OPEN și se specifică un URL constând din *mailto:* urmat de adresa destinatarului. Majoritatea programelor de navigare vor răspunde prin pornirea unei aplicații de poștă electronică cu adresa și câteva alte câmpuri din antet deja completate.

Protocolul *telnet* este utilizat pentru stabilirea unei conexiuni cu o mașină aflată la distanță. Se utilizează în același fel ca și programul telnet, ceea ce nu constituie o surpriză, deoarece majoritatea programelor de navigare utilizează programul telnet ca aplicație auxiliară.

Pe scurt URL-urile au fost proiectate nu numai pentru a permite utilizatorilor să navigheze prin Web, dar și pentru a utiliza FTP, news, Gopher, e-mail și telnet, ceea ce face inutile interfețele specializate pentru aceste protocoale integrând astfel într-un singur program, navigatorul în Web, aproape toate tipurile de acces în Internet. Dacă metoda nu ar fi fost proiectată de un fizician, ar fi putut să pară produsul departamentului de publicitate al unei companii de software.

În ciuda tuturor acestor proprietăți, creșterea Web-ului scoate în evidență și o slăbiciune a metodei utilizării URL-urilor. Pentru o pagină care este foarte des referită, ar fi de preferat să existe mai multe copii pe servere diferite, pentru a reduce traficul în rețea. Problema este că URL-rile nu oferă nici o posibilitate de indicare a unei pagini fără să se specifice unde este localizată pagina respectivă. Nu există nici o metodă pentru a spune ceva de genul: „Vreau pagina xyz, dar nu mă interesează de unde o aduci". Pentru a rezolva această problemă și a permite multiplicarea paginilor, IETF lucrează la un sistem de **URN** (**Universal Resource Names -** nume universale de resurse). Un URN poate să fie privit ca un URL generalizat. Acest subiect este în curs de cercetare, deși o propunere de sintaxă este dată în RFC 2141.

Lipsa stării și utilizarea cookies

Așa cum am văzut în mod repetat, Web-ul este, în principiu, lipsit de stare. Nu există conceptul unei sesiuni de conectare. Programul de navigare transmite o cerere către server și primește un fișier. Apoi serverul uită că a discutat vreodată cu acel client.

La început, când Web-ul a fost folosit doar pentru obținerea de documente accesibile publicului larg, acest model era perfect adaptat cerințelor. Dar, pe măsură ce Web-ul a început să capete și alte funcții, acest model a dat naștere unor probleme. De exemplu, anumite situri de Web impun clienților să se înregistreze (și chiar să plătească bani) spre a le utiliza. Ca atare, se pune întrebarea cum pot serverele să distingă între cereri din partea utilizatorilor înregistrați și a celorlalți. Un al doilea exemplu este comerțul electronic. Dacă un utilizator se plimbă printr-un magazin electronic, aruncând din când în când produse în coșul de cumpărături, cum poate serverul să rețină conținutul coșului? Un al treilea exemplu sunt portalurile de Web configurabile cum este Yahoo. Utilizatorii pot configura o pagină inițială detaliată, doar cu informația pe care o doresc (de ex.: valorile acțiunilor la bursă și echipele lor sportive favorite), dar cum poate serverul sa afișeze pagina corectă dacă nu știe cine este utilizatorul?

La o primă vedere, se poate crede că serverele pot să urmărească utilizatorii uitându-se la adresele lor IP. Această idee nu funcționează însă. În primul rând, mulți utilizatori lucrează pe calculatoare partajate cu alți utilizatori, în special în cadrul companiilor, iar adresa IP identifică doar calculatorul nu și utilizatorul. În al doilea rând, mult mai grav, mulți dintre cei ce oferă servicii de Internet (ISP) utilizează NAT, astfel încât toate pachetele ce pleacă de la orice utilizator folosesc aceeași adresă IP. Din punctul de vedere al serverului, cele câteva de mii de clienți ai unui ISP folosesc aceeași adresă IP.

Pentru a rezolva această problemă, Netscape a proiectat o tehnică mult criticată numită **cookies** (*rom. fursecuri*). Numele derivă dintr-un argou foarte vechi al programatorilor în care un program apelează o procedură și obține rezultate ce ar putea fi mai târziu pentru a executa ceva. În acest sens, o înregistrare de descriere a unui fișier UNIX sau un identificator al unui obiect Windows reprezintă un cookie. Mecanismul a fost formalizat mai târziu în RFC 2109.

Când un client cere o pagină de Web, serverul poate oferi informații adiționale odată cu pagina cerută. Aceste informații pot include un cookie, care este un fișier (sau șir de caractere) de dimensiune mică (cel mult 4 KB). Programele de navigare stochează cookie-urile oferite într-un director special pentru acestea pe discul clientului, cu excepția cazurilor când utilizatorul a oprit utilizarea cookie-urilor. Cookie-urile sunt doar fișiere sau șiruri de caractere, nu programe executabile. În principiu, un cookie ar putea conține un virus, dar deoarece cookie-urile sunt tratate ca date, nu există nici o posibilitate oficială ca virusul să fie executat și să cauzeze probleme. Este însă posibil ca un hacker să exploateze o eroare a programului de navigare și să cauzeze activarea virusului.

Un cookie poate conține până la cinci câmpuri, așa cum se arată în fig. 7-25. Câmpul *Domeniu* spune de unde a sosit cookie-ul. Programele de navigare trebuie să verifice că serverele nu mint în legătură cu domeniul lor. Fiecare domeniu poate stoca cel mult 20 de cookie-uri pentru fiecare client. Câmpul *Cale* reprezintă o cale în structura de directoare a serverului care identifică ce parte a arborelui de fișiere de pe server poate utiliza cookie-ul respectiv. Adesea, acest câmp este /, ceea ce înseamnă întregul arbore.

Domeniu	Cale	Conținut	Expiră	Sigur
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Da
joes-store.com	/	Cart1=1-00501;1-07031;2-13721	11-10-02 14:22	Nu
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	Nu
sneaky.com	/	UserID=3627239101	31-12-12 23:59	Nu

Fig. 7-25. Câteva exemple de cookie-uri

Câmpul *Conținut* are forma *nume* = *valoare*. Atât *nume* cât și *valoare* pot fi orice dorește serverul. Acesta este câmpul în care se stochează conținutul unui cookie.

Câmpul *Expiră* arată când expiră un cookie. Dacă acest câmp este absent, programul de navigare șterge cookie-ul la terminarea execuției. Un astfel de cookie se numește **cookie ne-persistent (non-persistent cookie)**. Dacă se oferă o dată și o oră, cookie-ul se numește **persistent** și este păstrat până la expirare. Timpul de expirare se dă pentru ora Greenwich (Greenwich Mean Time). Pentru a șterge un cookie de pe discul unui client, un server retransmite cookie-ul cu timpul de expirare în trecut.

În sfârșit, câmpul *Sigur* poate indica dacă programul de navigare poate transmite cookie-ul numai unui server sigur. Acest element este utilizat pentru comerțul electronic, aplicații bancare și alte aplicații sigure.

Am văzut cum se obțin cookie-urile, dar cum sunt ele utilizate? Chiar înainte ca un program de navigare să transmită cererea pentru o pagină către un sit Web, el verifică directorul de cookie-uri pentru a vedea dacă există vreun cookie care a fost stocat de către domeniul la care se duce cererea. În caz afirmativ, toate cookie-urile stocate de către acel domeniu sunt incluse în mesajul ce conține cererea. Atunci când serverul le obține, le poate interpreta în orice mod dorește.

Să examinăm acum câteva utilizări posibile pentru cookie-uri. În fig. 7-25, primul cookie a fost stocat de către *toms-casino.com* și este utilizat pentru a identifica utilizatorul. Când clientul se conectează săptămâna următoare pentru a arunca niște bani pe fereastră, programul de navigare transmite cookie-ul, astfel că serverul știe cine este clientul. Odată ce deține numărul de identificare al clientului, serverul poate căuta datele sale într-o bază de date și utiliza aceste informații pentru a construi o pagină de Web potrivită. Depinzând de obiceiurile clientului, această pagină poate reprezenta o masă de poker, o listă a curselor de cai din ziua respectivă sau o mașină de jocuri.

Cel de-al doilea cookie a venit de la *joes-store.com*. Scenariul în acest caz este că utilizatorul se plimbă prin magazin, căutând lucruri de cumpărat. Atunci când găsește un preț bun și execută un clic pe produsul respectiv, serverul construiește un cookie ce conține numărul de bucăți și codul produsului și îl transmite clientului. Pe măsură ce clientul continuă să se plimbe prin magazin, cookie-ul este întors la fiecare nouă pagină cerută. Pe măsură ce cumpărăturile se acumulează, serverul le adaugă la cookie. În figură, coșul de cumpărături conține trei produse, ultimul dintre ele în dublu exemplar. În cele din urmă, când clientul selectează *MERGI LA CASĂ*, cookie-ul, care acum conține lista completă de cumpărături este trimis odată cu cererea. În acest mod, serverul știe exact ce produse au fost cumpărate.

Cel de-al treilea cookie este pentru un portal de Web. Atunci când utilizatorul selectează o legătură către portal, programul de navigare transmite cookie-ul. Acesta spune portalului să construiască o pagină ce conține valorile acțiunilor pentru Sun Microsystems și Oracle și rezultatele echipei de fotbal New York Jets. Deoarece un cookie poate avea până la 4 KB, există suficient spațiu pentru preferințe mai detaliate în ceea ce privește titluri de articole din ziar, starea vremii, oferte speciale, ș.a.m.d.

Cookie-urile pot fi utilizate și în beneficiul serverului. De exemplu, să presupunem că un server dorește să știe în fiecare moment câți vizitatori a avut și câte pagini au fost vizitate de fiecare dintre aceștia înainte de a părăsi situl. Prima cerere nu va fi însoțită de nici un cookie, astfel că serverul va transmite înapoi un cookie ce conține *Counter=1*. Selecții ulterioare ale paginilor acestui site vor transmite acest cookie înapoi la server. De fiecare dată, contorul este incrementat și transmis înapoi clientului. Urmărind aceste contoare, serverul poate să vadă câte persoane renunță după vizitarea primei pagini, câte au vizitat două pagini, ș.a.m.d.

Cookie-urile au avut și utilizări greșite. Teoretic, cookie-urile ar trebui să ajungă doar la situl lor de origine, dar spărgătorii au exploatat numeroase erori în programele de navigare pentru a captura

cookie-uri care nu le erau destinate. Deoarece numeroase situri de comerț electronic pun numerele de cărți de credit în cookie-uri, potențialul pentru abuzuri este evident.

Un mod de utilizare controversat al cookie-urilor este colectarea, în secret, de informații privind obiceiurile de navigare pe Web ale utilizatorilor. Mecanismul funcționează în modul următor. O companie de publicitate, să spunem Sneaky Ads. (*rom. sneaky = viclean)* contactează un număr de situri de Web importante și pune anunțuri publicitare ale clienților săi pe paginile respectivelor situri, pentru care plătește celor ce dețin siturile o anumită sumă. În loc să ofere sitului un fișier GIF sau JPEG pentru a fi amplasat pe fiecare pagină, le oferă un URL ce trebuie adăugat la fiecare pagină. Fiecare din aceste URL-uri conține un număr unic în partea rezervată fișierului, cum ar fi

http://www.sneaky.com/382674902342.gif

Atunci când un utilizator vizitează o pagină *P* ce conține un asemenea anunț publicitar, programul de navigare obține fișierul HTML și vede legătura către imaginea de la *www.sneaky.com*, așa că transmite cererea pentru imagine acestui server. Serverul întoarce un fișier GIF conținând anunțul publicitar, împreună cu un cookie ce conține un număr unic de identificare pentru utilizator, 36271239101 în fig. 7-25. Compania Sneaky înregistrează faptul că utilizatorul cu acest număr de înregistrare a vizitat pagina *P.* Aceasta este ușor de realizat având în vedere faptul că fișierul cerut (*382674902342.gif*) este menționat doar în pagina *P*. Desigur că anunțul în sine poate apare pe o mie de alte pagini, dar de fiecare dată cu un nume de fișier diferit. Compania Sneaky colectează probabil doar câțiva bănuți de la compania ce a realizat produsul de fiecare dată când transmite anunțul publicitar.

Mai târziu, când utilizatorul vizitează o altă pagină de Web care conține unul din anunțurile publicitare ale companiei Sneaky, după ce programul de navigare a obținut fișierul HTML de la server, vede referința către, să spunem, *http://www.sneaky.com/493654919923.gif* și cere acest fișier. Deoarece există deja un cookie de la domeniul *sneay.com*, programul de navigare include cookie-ul companiei Sneaky ce conține și numărul unic de identificare al utilizatorului. Compania Sneaky știe acum o a doua pagină vizitată de utilizator.

Cu timpul, compania Sneaky poate construi un profil complet al obiceiurilor de navigare ale utilizatorului, deși acesta nu a efectuat nici un clic pe vreun anunț publicitar. Desigur, acest profil nu include încă numele utilizatorului (deși conține adresa IP a acestuia, informație care poate fi suficientă pentru a deduce numele din alte baze de date). În cazul în care utilizatorul oferă vreodată numele său unui site care cooperează cu Sneaky, un profil complet ce include și numele este acum gata de vânzare pentru oricine dorește să-l cumpere. Vânzarea acestor informații poate fi suficient de profitabilă pentru ca Sneaky sa poată plasa mai multe anunțuri publicitare pe mai multe situri Web și astfel să colecteze și mai multe informații. Cea mai ascunsă parte a acestei povești este că majoritatea utilizatorilor nu știu nimic despre această colectare de informații și chiar ar putea să se creadă în siguranță, pentru că nu au selectat nici un anunț publicitar.

Și dacă Sneaky vrea să fie și mai vicleană, anunțul publicitar poate să nu fie unul clasic. Un „anunț" format dintr-un singur pixel de culoarea fondului (și deci invizibil) are exact același efect ca și anunțul propriu-zis: programul de navigare trebuie să obțină imaginea gif de 1 x 1 pixeli și să îi livreze toate cookie-urile care au fost transmise de domeniul de origine al pixelului.

Pentru a menține impresia de intimitate, o serie de utilizatori își configurează programele de navigare pentru a refuza toate cookie-urile. Această acțiune poate duce însă la probleme cu siturile de Web legitime ce folosesc cookie-uri. Pentru a rezolva această problemă, utilizatorii instalează câteodată software care „mănâncă cookie-uri" (*cookie-eating software*). Acestea sunt programe speciale care inspectează fiecare cookie la sosire și îl acceptă sau refuză în funcție de opțiunile utilizatorului

(de ex.: în ce situri Web se poate avea încredere). Această metodă oferă utilizatorului un control fin asupra căror cookie-uri sunt acceptate și care sunt refuzate. Programele de navigare moderne cum ar fi Mozilla (*www.mozilla.org*) au un nivel de control complex asupra cookie-urilor.

7.3.2 Documente Web statice

Fundamentul Web-ului este transferul de pagini de Web de la server la client. În forma cea mai simplă, paginile de Web sunt statice, adică doar fișiere existente pe un server, ce așteaptă să fie cerute. În acest sens, chiar și o înregistrare video este o pagină de Web statică, deoarece este doar un fișier. În această secțiune vom privi paginile de Web statice în detaliu. În secțiunea următoare vom examina conținutul dinamic.

HTML--HyperText Markup Language

Paginile de Web sunt în prezent scrise într-un limbaj numit **HTML (HyperText Markup Language)**. HTML permite utilizatorilor să producă pagini de Web care conțin text, imagini și referințe la alte pagini de Web. HTML este un limbaj de marcare, un limbaj care descrie cum trebuie să fie formatate textele. Termenul de „marcare" provine din timpurile vechi, când editorii făceau marcaje pe documente pentru a indica tipografului - în acele timpuri un om - ce font-uri să folosească ș.a.m.d. Limbajele de marcare conțin comenzi explicite pentru formatare. De exemplu, în HTML, **înseamnă început de mod aldin, și** înseamnă terminarea utilizării modului aldin. Avantajul utilizării unui limbaj de marcare față de unul în care nu se utilizează marcarea explicită constă din faptul că este simplu de scris un program de navigare care să interpreteze comenzile de marcare. TeX și troff sunt două exemple foarte cunoscute de limbaje de marcare.

Prin standardizarea și includerea comenzilor de marcaj în fiecare fișier HTML, devine posibil ca orice program de navigare să poată să citească și să formateze orice pagină Web. Posibilitatea formatării paginii recepționate este foarte importantă, deoarece o pagină poate să fie construită pe un ecran cu 1600 x 1200 pixeli utilizând culori codificate cu 24 de biți, dar s-ar putea să fie necesară afișarea într-o mică fereastră de pe un ecran cu 640 x 320 pixeli și utilizând culori codificate pe 8 biți.

În cele ce urmează se face o scurtă introducere în limbajul HTML, doar pentru a oferi o idee despre subiect. Cu toate că este posibil să se construiască documente HTML utilizând orice editor, și mulți fac asta, este posibil și să se utilizeze editoare HTML speciale care pot să facă toată munca (desigur, în mod corespunzător utilizatorul are mai puțin control asupra detaliilor produsului final).

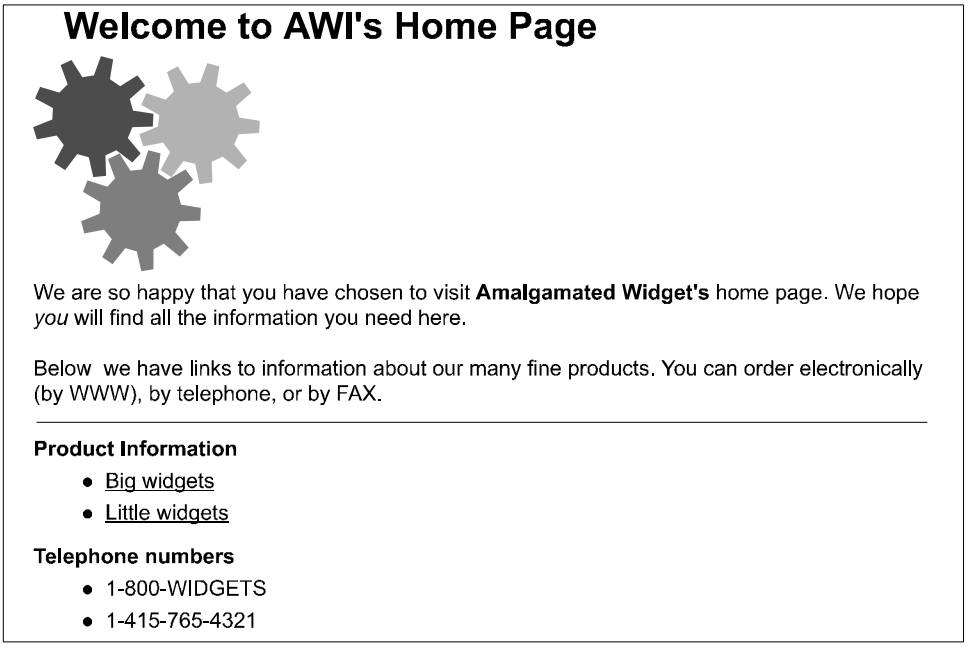
O pagină Web corect formată conține o zonă de cap și o zonă de corp, cuprinse între marcajele (tag-uri) <html> și </html>, dar majoritatea programelor de navigare ignoră absența acestor marcaje. Așa cum se vede în fig. 7-26(a), capul este cuprins între marcajele <head> și </head>, iar corpul între marcajele <body> și </body>. Comenzile cuprinse între aceste marcaje se numesc **directive**. Majoritatea marcajelor HTML au acest format, adică, <ceva> pentru a indica începutul a ceva și </ceva> pentru a marca sfârșitul. Numeroase exemple de fișiere HTML sunt disponibile. Majoritatea programelor de navigare au o opțiune VIEW SOURCE (afișarea sursei) sau ceva similar. Selectarea acestei opțiuni afișează pagina curentă în format HTML în loc de forma interpretată.

Marcajele pot să fie scrise cu litere mici sau mari. Adică <head> și <HEAD> înseamnă același lucru, dar versiuni mai noi ale standardului cer existența doar a primei forme. Cum este dispus textul în documentul HTML este nesemnificativ. Programele de navigare ignoră spațiile și trecerile la rând nou, deoarece textul trebuie să fie formatat pentru a corespunde zonei de afișare curente. Cores-

punzător, se pot utiliza spații pentru a face documentele HTML mai ușor de citit, ceva ce ar fi necesar pentru majoritatea documentelor. Liniile albe nu pot să fie utilizate pentru separarea paragrafelor, deoarece sunt pur și simplu ignorate. Este necesară utilizarea unor marcaje explicite.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title></head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p> Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by FAX. </p>
<hr>
<h2> Product Information </h2>
<ul>
        <li> <a href="http://widget.com/products/big" > Big widgets </a>
        <li> <a href="http://widget.com/products/little" > Little widgets </a>
</ul>
<h2> Telephone Numbers </h2>
<ul>
        <li> By telephone: 1-800-WIDGETS,
        <li> By fax: 1-415-765-4321
</ul>
</body>
</hmtl>
```

(a)


Welcome to AWI's Home Page
We are so happy that you have chosen to visit Amalgamated Widget's home page. We hope you will find all the information you need here.
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.
Product Information
Big widgets
Little widgets
Telephone numbers
1-800-WIDGETS
1-415-765-4321

(b)

Fig.7-26. (a) Un exemplu simplu de pagină de Web. (b) Pagina formatată.

Unele marcaje au parametri (care au nume), numiți **atribute**. De exemplu:

```
<img src="abc" alt="foobar">
```

este un marcaj, <img>, având parametrul *src* cu valoarea *abc* și parametrul *alt* cu valoarea *foobar*. Pentru fiecare marcaj, standardul HTML oferă o listă a parametrilor care pot să fie utilizați, dacă este cazul și care este semnificația lor. Deoarece parametrii au nume, ordinea în care se dau valorile parametrilor nu este semnificativă.

Din punct de vedere tehnic, documentele HTML sunt scrise utilizând setul de caractere ISO 8859-1 Latin-1, dar pentru utilizatorii ale căror tastaturi suportă numai codul ASCII, se pot utiliza secvențe de caractere pentru reprezentarea caracterelor speciale cum ar fi è. Lista caracterelor speciale este precizată în standard. Toate încep cu caracterul „&” și se termină cu „;”. De exemplu è produce è iar é produce é. Deoarece <, > și & au semnificații speciale, pot să fie reprezentate numai utilizând secvențele speciale de caractere corespunzătoare, < > și &.

Principalul element din zona de cap este titlul care este cuprins între <title> și </title>, dar aici pot să apară și alte tipuri de informații. Titlul nu este afișat pe pagină. Unele programe de navigare îl utilizează pentru a eticheta fereastra în care se afișează pagina respectivă.

Să analizăm și alte particularități prezente în exemplul din fig. 7-26. Toate marcajele utilizate în fig. 7-26 și încă alte câteva sunt prezentate în fig. 7-27. Titlurile de capitole sunt generate de marcajul <h*n*>, unde *n* este o cifră între 1 și 6. <h1> este titlul cel mai important; <h6> este cel mai puțin important.

Marcaj	Descriere
<html> ... </html>	Delimitează textul scris în HTML
<head> ... </head>	Delimitează zona de cap
<title> ... </title>	Definește titlul (nu este afișat de programul de navigare)
<body> ... </body>	Delimitează zona de corp
<h*n*> ... </h*n*>	Delimitează un titlu de nivel *n*
...	Textul ... o să fie afișat cu aldine
...	Textul ... o să fie afișat cu cursive
...	Centrează ... pe pagină orizontal
...	Delimitează o listă neordonată
...	Delimitează o listă ordonată (numerotată)
...	Delimitează un elemente într-o listă ordonată sau neordonată
 	Trecere la linie nouă
<p>	Început de paragraf
<hr>	Linie orizontală
<img src=„...”>	Se încarcă o imagine
 ...	Se definește o hiper-legătură

Fig. 7-27. O selecție de marcaje uzuale. Unele mai au și alți parametri.

Depinde de programul de navigare să prezinte aceste titluri în mod diferit pe ecran. De obicei, titlurile cu număr mai mic vor fi afișate utilizând caractere mai mari. Programul de navigare poate să utilizeze culori diferite pentru fiecare nivel de titlu. De obicei, pentru titlurile marcate cu <h1> se utilizează litere mari scrise cu aldine cu cel puțin o linie liberă înainte și după. Corespunzător, pentru titlurile marcate cu <h2>, se utilizează caractere mai mici cu mai puțin spațiu lăsat înainte și după.

Marcajele și sunt utilizate pentru a indica modurile aldin și respectiv cursiv. Dacă programul de navigare nu poate să afișeze aceste tipuri de caractere, va utiliza un alt mod de a le reprezenta, de exemplu utilizând diferite culori sau video-invers.

Limbajul HTML oferă diferite mecanisme pentru construirea de liste, inclusiv liste conținute în alte liste. Listele încep cu <ul> sau <ol>, <li> fiind folosit pentru a marca începutul elementelor în ambele cazuri. Marcajul <ul> indică începutul unei liste neordonate. Elementele individuale, care sunt marcate în sursă cu <li>, sunt reprezentate precedate de buline (•). O variantă a acestui mecanism este <ol>, care descrie o listă ordonată. Când se utilizează acest marcaj, textele precedate de <li> sunt numerotate de către programul de navigare. Marcajele <ul> și <ol> au aceeași sintaxă și efecte asemănătoare.

Marcajele <br>, și <hr> indică o separare între diferitele părți ale textului. Formatul precis este descris în pagina de stil (vezi mai jos) asociată cu pagina curentă. Marcajul <br> forțează trecerea la linie nouă. De obicei programele de navigare nu inserează o linie liberă după <br>. Marcajul reprezintă un început de paragraf, pentru care se va insera o linie nouă și eventual se va face o indentare. (În mod teoretic, există și pentru a indica sfârșitul de paragraf, dar este foarte rar utilizat; majoritatea celor care scriu în HTML nici nu știu că acest marcaj există). Ultimul marcaj, <hr>, forțează trecerea la linie nouă și desenează o linie orizontală.

HTML permite includerea de imagini în paginile de Web. Marcajul <img> arată că pe poziția curentă din pagină se va include o imagine. Marcajul poate să aibă o serie de parametri. Parametrul *src* indică URL-ul imaginii. Standardul HTML nu specifică ce formate grafice sunt permise. În practică, toate programele de navigare acceptă fișiere în format GIF, multe pot lucra și cu fișiere în format JPEG. Programele de navigare pot să lucreze și cu alte formate, dar o astfel de extensie este o sabie cu două tăișuri. Dacă, de exemplu un utilizator este obișnuit cu un program de navigare care suportă fișiere în format BMP, este posibil ca el să includă astfel de fișiere în pagina sa de Web și să fie uimit că alte programe de navigare ignoră imaginile sale minunate.

Alți parametri pentru <img> sunt *align*, care controlează modul în care se aliniază imaginea față de limita de jos a textului (*top, middle, bottom*), *alt*, care furnizează textul afișat în locul imaginii dacă utilizatorul dezactivează opțiunea de afișare a imaginilor și *ismap*, un indicator care anunță că imaginea este o hartă selectabilă.

În sfârșit, să considerăm hiper-legăturile, care utilizează marcajele <a> (anchor) și . Ca și <img>, <a> are diverși parametrii, printre care *href* (URL-ul) și *name* (numele hiper-legăturii). Textul cuprins între <a> și este afișat. Dacă este selectat, atunci se utilizează hiper-legătura pentru a se aduce o nouă pagină. În locul textului se poate pune și un marcaj <img>, caz în care, cu un clic pe imagine, se va activa legătura.

De exemplu, să considerăm următorul fragment HTML:

```
<a href=„http://www.nasa.gov”>NASA’s home page </a>
```

Când se afișează acest fragment, pe ecran apare:

NASA’s home page

Dacă utilizatorul execută un clic pe acest text, programul de navigare aduce și afișează pagina al cărei URL este *http://www.nasa.gov*.

Ca al doilea exemplu, să considerăm

```
<a href=„http://www.nasa.gov”><img src=„shuttle.gif”alt=„NASA”></a>
```

Când se afișează pagina va apărea o imagine (o navetă spațială). Executând un clic pe imagine, se va aduce pagina NASA, la fel ca și în cazul în care în exemplul anterior a fost selectat textul. Dacă utilizatorul a dezactivat opțiunea de afișare a imaginilor, atunci în loc de imagine va fi afișat textul NASA.

```
<html>
<head><title> A sample page with a table </title></head>
<body>
<table border=1 rules=all>
<caption> Some differences between HTML Versions </caption>
<col align=left>
<col align=center>
<col align= center >
<col align= center >
<col align= center >
<tr> <th>Item <th>HTML 1.0 <th>HTML 2.0 <th>HTML 3.0 <th> HTML 4.0 </tr>
<tr> <th> Hyperlinks <td> x <td> x <td> x <td> x </tr>
<tr> <th> Images <td> x <td> x <td> x <td> x </tr>
<tr> <th> Lists <td> x <td> x <td> x <td> x </tr>
<tr> <th> Active Maps and Images <td>   <td> x <td> x <td> x </tr>
<tr> <th> Forms <td>   <td> x <td> x <td> x </tr>
<tr> <th> Equations <td>   <td>   <td> x <td> x </tr>
<tr> <th> Toolbars <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tables <td>   <td>   <td> x <td> x </tr>
<tr> <th> Accesibility features <td>   <td>   <td>   <td> x </tr>
<tr> <th> Object embedding <td>   <td>   <td>   <td> x </tr>
<tr> <th> Scripting <td>   <td>   <td>   <td> x </tr>
</table>
</body>
</html>
```

(a)

Some diferences between HTML Versions

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Hyperlinks	x	x	x	x
Images	x	x	x	x
Lists	x	x	x	x
Active Maps and Images		x	x	x
Forms		x	x	x
Equations			x	x
Toolbars			x	x
Tables			x	x
Accessibility features				x
Object embedding				x
Scripting				x

(b)

Fig. 7-28. (a) O tabelă HTML, (b) Un rezultat posibil.

Pentru marcajul <a> se poate utiliza parametrul *name* pentru a fixa o hiper-legătură, care să fie referită din pagină. De exemplu, unele pagini de Web încep cu o tabelă de conținut selectabilă. Prin execuția unui clic pe o intrare în tabela de conținut, se va trece direct la secțiunea corespunzătoare din pagină.

HTML evoluează continuu. În versiunile HTML 1.0 și HTML 2.0 nu existau tabele, dar au fost adăugate în HTML 3.0. O tabelă HTML este formată din una sau mai multe linii, fiecare fiind formată din una sau mai multe **celule**. Celulele pot să conțină diferite tipuri de informații, inclusiv text, figuri, iconițe, fotografii și chiar tabele. Celulele pot să fie alipite, de exemplu un titlu poate să se întindă peste mai multe coloane. Autorii paginilor au control asupra modului în care se face afișarea, inclusiv alinierea, stilul bordurii, marginile celulelor, dar programul de navigare este cel care hotărăște de fapt cum se face afișarea.

O descriere în HTML a unei tabele este prezentată în fig. 7-28(a), iar efectul posibil este prezentat în fig. 7-28(b). Acest exemplu prezintă câteva din facilitățile de descriere a tabelelor în HTML. Tabelele încep cu marcajul <table>. Se pot specifica informații suplimentare pentru a descrie proprietățile generale ale tabelei.

Marcajul <caption> poate să fie utilizat pentru a furniza un titlu tabelei. Fiecare linie începe cu marcajul <tr> (Table Row - linie în tabelă). Celulele individuale sunt marcate cu <th> (Table Header - titlu de coloană), <td> (Table Data - date în tabelă). Diferențierea este necesară pentru a permite programului de navigare să le afișeze diferit, așa cum se vede și din exemplul considerat.

În tabele se pot utiliza alte marcaje. Acestea includ posibilitatea de a specifica alinieri orizontale sau verticale ale celulelor, alinierea în cadrul celulei, margini, gruparea de celule, unități și multe altele.

În HTML 4.0 au fost adăugate noi elemente. Acestea includ elemente ce fac paginile mai accesibile utilizatorilor cu handicap, înglobarea obiectelor (o generalizare a marcajului <img> astfel încât alte obiecte să poată fi înglobate în pagini), suport pentru limbaje de scripturi (pentru a permite conținut dinamic) și multe altele.

Când un sit de Web este complex, fiind format din multe pagini produse de autori diferiți ce lucrează pentru aceeași companie, este adesea de dorit să existe o modalitate pentru a împiedica moduri de prezentare diferite în pagini diferite. Această problemă poate fi rezolvată utilizând **paginile de stil (style sheets)**. Atunci când se utilizează pagini de stil, paginile individuale nu mai folosesc stiluri fizice, cum sunt modurile aldin și cursiv. Autorii pot acum să utilizeze stiluri logice, cum sunt <dn> (definiție), (evidențiere), (evidențiere accentuată) și <var> (variabile de program). Stilurile logice sunt definite în pagina de stil, pentru care există o referință la începutul fiecărei pagini. În acest fel, toate paginile au același stil și dacă administratorul sitului (*Webmaster*) decide să schimbe stilul din stil cursiv de 14 puncte tipografice, culoare albastră în stil aldin, 18 puncte tipografice, culoare roz țipător, tot ceea ce trebuie să facă este să schimbe o singură definiție pentru a converti întregul sit Web. O pagină de stil poate fi comparată cu o directivă #include într-un program C: schimbarea unei macrodefiniții în fișierul inclus determină schimbarea în toate fișierele program ce includ respectivul header.

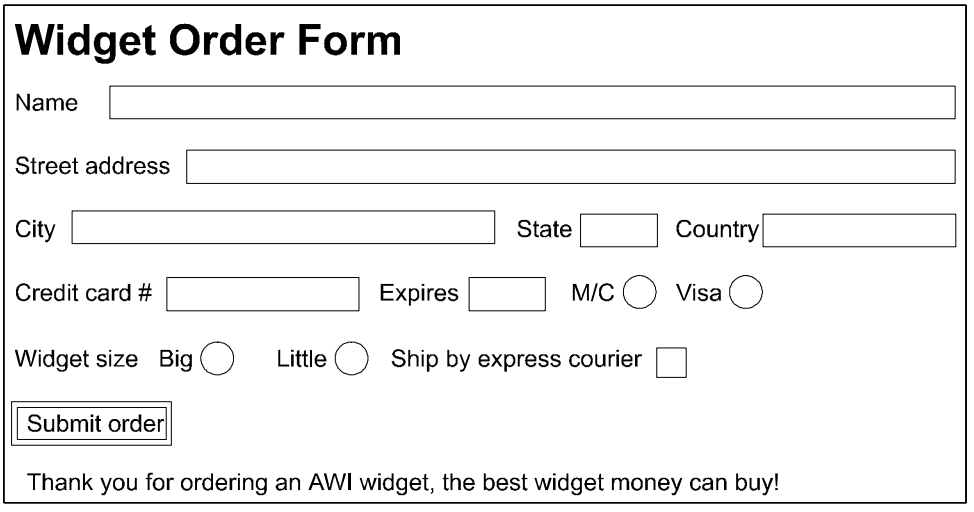
Formulare

HTML 1.0 funcționa într-o singură direcție. Utilizatorii puteau să aducă o pagină de la furnizorii de informație, dar era foarte dificil să se transmită informație în sens invers. Pe măsură ce tot mai multe organizații comerciale au început să utilizeze Web-ul, a apărut o puternică cerere pentru comunicația în dublu sens. De exemplu, multe companii vor să poată prelua comenzi pentru produse utilizând paginile lor de Web, furnizorii de software vor să distribuie programe prin intermediul Web-ului, clienții să își completeze fișele de înregistrare prin același mijloc, iar companiile care oferă servicii de căutare în Web au nevoie ca utilizatorii de servicii să poată să introducă cuvintele pe baza cărora se face căutarea.

Acest gen de cereri a dus la includerea **formularelor** începând cu HTML 2.0. Formularele conțin casete și butoane care permit utilizatorilor să completeze informații sau să facă selecții și apoi să transmită informațiile la proprietarul paginii. În acest scop se utilizează marcajul <input>. Acesta are o varietate de parametri care determină mărimea, tipul, și modul de afișare a casetei utilizate. Cele mai obișnuite sunt câmpuri în care utilizatorul poate să introducă text, casete care pot să fie selectate, hărți active, butoane *submit*. Exemplul din fig. 7-29 prezintă câteva dintre aceste posibilități.

```
<html>
<head><title> AWI CUSTOMER ORDERING FORM </title></head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size=4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p> <input type=submit value="submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

(a)



(b)

Fig. 7-29. (a) Un formular de comandă HTML. (b) Pagina formatată.

Să începem discuția parcurgând exemplul. Ca orice formular, și acesta este cuprins între marcajele <form> și </form>. Textele care nu sunt incluse între marcaje sunt afișate. Într-un formular poate să fie utilizat orice marcaj obișnuit (de exemplu <b>) În acest formular sunt utilizate trei tipuri de casete.

Prima casetă din formular apare după textul „Name". Caseta are lățimea de 46 de caractere și utilizatorul va introduce un șir care va fi memorat în variabila *customer* pentru prelucrări ulterioare. Marcajul <p> indică programului de navigare să afișeze ceea ce urmează pe o linie nouă, chiar dacă mai este loc pe linia curentă. Utilizând <p> și alte marcaje care controlează dispunerea textului, creatorul paginii poate să controleze cum arată formularul pe ecran.

Pe linia următoare se solicită adresa utilizatorului, având cel mult 40 de caractere, pe o linie separată. Urmează o linie pe care se solicită orașul, statul și țara. Aici nu se utilizează marcajul <p>, deci programul de navigare o să le afișeze pe toate pe aceeași linie, dacă încap. Din punctul de vedere al programului de navigare, paragraful curent conține șase elemente: trei șiruri alternând cu trei casete. El le afișează pe aceeași linie de la stânga la dreapta, trecând la o linie nouă ori de câte ori pe linia curentă nu mai încape următorul element. Astfel, este posibil ca pe un ecran 1600 x 1200 să încapă toate cele trei șiruri și casetele corespunzătoare, în timp ce pe un ecran 1024 x 768 ele pot să fie distribuite în două linii. În cazul cel mai defavorabil cuvântul „Country" este la capătul liniei, iar caseta asociată este la începutul liniei următoare. Nu există nici o posibilitate de a forța programul de navigare să afișeze caseta lângă text.

Următoarea linie solicită numărul cărții de credit și data sa de expirare. Transmiterea numerelor cărților de credit prin Internet trebuie să se facă numai dacă s-au luat măsurile de securitate adecvate. Vom discuta despre aceste măsuri în cap. 8.

După data de expirare, întâlnim un element nou: butoane radio. Acestea sunt utilizate atunci când trebuie să se facă o alegere între mai multe alternative. Modelul care se utilizează aici este cel al unui aparat de radio care are butoane pentru selecția scalelor. Programul de navigare afișează aceste casete într-o formă care permite utilizatorului să le selecteze sau să le deselecteze prin execuția unui clic (sau utilizând tastatura). Selecția unuia dintre ele le deselectează pe toate celelalte care fac parte din același grup. Modul de afișare depinde de programul de navigare. „Widget size" utilizează de asemenea două butoane. Cele două grupuri sunt diferențiate prin câmpul *name* și nu printr-un domeniu static de genul <radiobutton> ... </radiobutton>.

Parametrii *value* sunt utilizați pentru a arăta care buton a fost apăsat. În funcție de opțiunea aleasă pentru cartea de credit, variabila *cc* va avea ca valoare șirul „mastercard" sau „visacard".

După cele două seturi de butoane, urmează opțiunea referitoare la modul de transport, reprezentată de o casetă de tip *checkbox*. Aceasta poate să fie pe poziția selectat sau nu. Spre deosebire de cazul butoanelor radio, unde poate să fie selectat un singur buton dintr-un set, fiecare casetă de tip *checkbox* poate să fie selectată sau nu, independent de celelalte. De exemplu, când se comandă o pița utilizând pagina de Web Electropizza, utilizatorul poate să aleagă sardele *și* ceapă *și* ananas (dacă le suportă), dar nu poate să aleagă mică *și* medie *și* mare pentru aceeași pița. Conținutul corespunzător piței va fi reprezentat de trei butoane diferite de tip *checkbox,* în timp ce dimensiunea va fi reprezentată de un set de butoane radio.

Pe de altă parte, în cazul în care lista din care se face alegerea este foarte lungă, butoanele radio devin dificil de utilizat. Din acest motiv, marcajele <select> și </select> sunt utilizate pentru a prezenta o listă de alternative, utilizând semantica corespunzătoare unor butoane radio (dacă nu se utilizează parametrul *multiple*, caz în care semantica este cea de la casetele de tip

checkbox). Unele programe de navigare afișează opțiunile cuprinse între <select> și </select> ca un meniu derulant.

Am văzut jumătate din tipurile standard pentru marcajul <input>: *radio* și *checkbox*. De fapt, am văzut și un al treilea: *text*. Deoarece acesta este tipul implicit, nu am mai utilizat parametrul *type = text*, dar puteam să o facem. Alte două tipuri sunt *password* și *textarea*. O casetă *password* funcționează la fel ca o casetă *text*, numai că nu se face afișarea caracterelor introduse. O casetă *textarea* este similară unei casete *text*, numai că va conține mai multe linii.

Întorcându-ne la exemplul din fig. 7-29, urmează butonul *submit*. Când este selectat acest buton, informația introdusă de către utilizator este transmisă la calculatorul de pe care provine formularul. Similar altor tipuri, *submit* este un cuvânt cheie pe care îl înțelege programul de navigare. Șirul *value* reprezintă în acest caz eticheta butonului și se afișează. Toate casetele pot să aibă valori, dar am avut nevoie de această facilitate numai aici. Pentru casetele *text*, conținutul câmpului *value* este afișat o dată cu formularul, dar utilizatorul poate să îl modifice sau să îl șteargă. Casetele *checkbox* și *radio* pot să fie inițializate, utilizând însă un parametru numit *checked* (deoarece parametrul *value* oferă un text, dar nu indică o selecție).

Atunci când utilizatorul selectează butonul „submit order”, programul de navigare împachetează informația colectată într-o singură linie lungă și o transmite serverului pentru procesare. Pentru a separa câmpurile, se utilizează caracterul &; caracterul + reprezintă spațiu. De exemplu, răspunsul la formular poate să arate ca în fig. 7-30:

```
(împărțit aici în trei linii din motive de aliniere pagină)

customer=John+Doe&address=100+Main+St.&city=White+Plain&
state=NY&country=USA&cardno=1234567890&expires6/98&cc=mastercard&
product=cheap&express=on
```

Fig. 7-30. Un răspuns posibil de la programul de navigare către server cu informațiile completate de utilizator

Șirul va fi transmis la server ca o singură linie, nu ca trei. Dacă un *checkbox* nu este selectat, el este omis din șir. Este problema serverului să interpreteze șirul respectiv. Vom discuta cum se poate realiza acest lucru mai târziu în acest capitol.

XML și XSL

Limbajul HTML, cu sau fără formulare, nu conferă nici o structură paginilor de Web. De asemenea, el amestecă conținutul cu informații despre formatul paginii. Pe măsură ce comerțul electronic și alte aplicații devin din ce în ce mai răspândite, apare o cerere din ce în ce mai mare pentru structurarea paginilor de Web și separarea conținutului de informațiile de format. De exemplu, un program care caută pe Web prețul cel mai bun al unei cărți sau al unui CD trebuie să analizeze multe pagini de Web căutând numele produsului și prețul său. Cu paginile de Web în format HTML este foarte dificil ca un program să își dea seama unde se află numele și unde se află prețul.

Din acest motiv, consorțiul W3C a dezvoltat îmbunătățiri ale limbajului HTML pentru a permite paginilor de Web să fie structurare în vederea procesării automate. Două limbaje noi au fost dezvoltate în acest scop. Mai întâi, **XML (eXtensible Markup Language)** descrie conținutul într-un mod

structurat și apoi, **XSL (eXtensible Style Language)** descrie formatul independent de conținut. Ambele limbaje reprezintă subiecte mari și complexe, ca atare scurta introducere care urmează atinge tangențial subiectul, deși ar trebui să ofere o idee despre modul cum funcționează.

Să considerăm documentul XML dat ca exemplu în fig. 7-31. Acesta definește o structură numită book_list, care reprezintă o listă de cărți. Fiecare carte (book) are trei câmpuri, titlul, autorul și anul publicării. Aceste structuri sunt extrem de simple. Sunt permise structurile ce conțin câmpuri repetitive (de ex.: mai mulți autori), câmpuri opționale (de ex.: titlul CD-ROM-ului inclus) și câmpuri la alegere (de ex.: URL-ul unei librării dacă respectiva carte mai este disponibilă sau URL-ul unui sit de licitații dacă nu mai este disponibilă pe piață).

În acest exemplu, fiecare din cele trei câmpuri este o entitate indivizibilă, dar se permite subdivizarea ulterioară a unui câmp. De exemplu, câmpul autor putea fi alcătuit așa cum urmează, spre a oferi un control mai fin la căutare și formatare:

```
<author>
        <first_name>Andrew</first_name>
        <last_name>Tannenbaum</last_name>
</author>
```

Fiecare câmp poate fi împărțit în sub-câmpuri și sub-sub-câmpuri până la o adâncime arbitrară.

Întregul fișier din fig. 7-31 definește o listă de cărți ce conține trei cărți. Nu spune nimic despre modul cum se poate afișa pagina de Web pe ecran. Pentru a oferi informații de formatare avem nevoie de un al doilea fișier, *book_list.xsl*, ce conține definiția XSL. Acest fișier este o pagină de stil care spune cum se poate afișa pagina. (Există alternative la paginile de stil, cum ar fi conversia XML în HTML, dar aceste alternative depășesc subiectul acestei cărți.)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>

<book>
        <title>Computer Networks, 4/e </title>
        <author> Andrew S. Tannenbaum </author>
        <year> 2003 </year>
</book>

<book>
        <title> Modern Operating Systems, 2/e </title>
        <author> Andrew S. Tannenbaum </author>
        <year> 2001 </year>
</book>

<book>
        <title> Structured Computer Organization 4/e </title>
        <author> Andrew S. Tannenbaum </author>
        <year> 1999 </year>
</book>

</book_list>
```

Fig. 7-31. O pagină de Web simplă în format XML

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">

<html>
<body>

<table border="2">
 <tr>
 <th> Title </th>
 <th> Author </th>
 <th> Year </th>
 </tr>

 <xsl:for-each select="book_list/book">
 <tr>
 <td> <xsl: value-of select="title"/> </td>
 <td> <xsl: value-of select="author"/> </td>
 <td> <xsl: value-of select="year"/> </td>
 </tr>
 </xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Fig. 7-32. O pagină de stil în XSL

Un exemplu de fișier XSL pentru formatarea conținutului din fig. 7-31 este dat în fig. 7-32. După câteva declarații necesare ce includ URL-ul standardului XSL, fișierul conține marcaje începând cu <html> și <body>. Acestea definesc începutul unei pagini de Web, ca de obicei. Urmează apoi o definiție de tabel ce include titlurile celor trei coloane. Să observăm că în plus față de marcajele <th> există și marcaje </th>, lucru care nu ne-a preocupat până acum. Specificațiile XML și XSL sunt mult mai stricte decât specificația HTML. Ele statuează că rejectarea fișierelor incorecte din punct de vedere sintactic este obligatorie, chiar dacă programul de navigare poate determina ce a intenționat proiectantul paginii Web. Un program de navigare care acceptă fișiere XML sau XSL incorecte din punct de vedere sintactic și repară eroarea el însuși este neconform cu standardul și va fi rejectat la un test de conformanță cu standardele. Programelor de navigare li se permite însă să identifice exact eroarea. Această măsură întrucâtva draconică este necesară pentru a rezolva problema numărului imens de pagini de Web scrise neglijent, care există în prezent.

Instrucțiunea

```
<xsl: for-each select="book_list/book">
```

este comparabilă cu o instrucțiune *for* în C. Execuția ei determină programul de navigare să execute corpul buclei (terminată de <xsl:for-each>) câte o dată pentru fiecare carte. Fiecare iterație afișează cinci linii: <tr>, titlul, autorul și anul și </tr>. După această buclă, sunt transmise la ieșire marcajele de închidere <body> și </html>. Rezultatul operației programului de navigare de a interpreta această pagină de stil este același ca și în cazul când pagina de Web ar fi conținut direct tabelul. Totuși, în acest format, programele pot analiza fișierul XML și găsi cu ușurință cărțile publicate du-

pă 2000, de exemplu. Merită subliniat faptul că deși fișierul nostru XSL conținea un fel de buclă, paginile de Web în XML și XSL sunt în continuare statice deoarece conțin pur și simplu instrucțiuni pentru programul de navigare despre modul de afișare a paginii, la fel ca și paginile HTML. Desigur, pentru a folosi XML și XSL, programul de navigare trebuie să fie capabil să interpreteze XML și XSL, dar marea majoritate a acestora au deja această capacitate. Nu este încă foarte clar dacă XSL va prelua paginile de stil tradiționale.

Nu am arătat cum se poate face acest lucru, dar limbajul XML permite proiectantului de situri Web să creeze fișiere de definiție în care structurile sunt definite în avans. Aceste fișiere de definiții pot fi incluse unele în altele, făcând posibilă construcția de pagini Web complexe. Pentru informații suplimentare despre aceasta și multe alte caracteristici ale limbajelor XML și XSL, consultați una din multele cărți despre acest subiect. Două exemple sunt (Livingston, 2002; și Williamson, 2001).

Înainte de a încheia discuția despre XML și XSL, merită să comentăm pe marginea luptei ideologice din interiorul consorțiului WWW și a comunității dezvoltatorilor de pagini Web. Scopul inițial al limbajului HTML era să specifice *structura* documentului și nu *modul de afișare*. De exemplu,

```
<h1> Deborah's Photos </h1>
```

instruiește programul de navigare să sublinieze titlul, dar nu spune nimic despre tipului font-ului, dimensiune sau culoare. Acestea sunt lăsate pe seama programului de navigare, care cunoaște proprietățile ecranului (de ex.: câți pixeli are). Totuși, mulți dezvoltatori de pagini Web doreau controlul absolut asupra modalității în care erau afișate paginile lor, astfel că au fost adăugate noi marcaje la HTML pentru a controla modul de afișare, cum ar fi

```
<font face="helvetica" size="24" color="red"> Deborah's Photos </font>
```

De asemenea, au fost adăugate modalități de a controla poziționarea precisă pe ecran. Această abordare este că nu este portabilă, ceea ce reprezintă desigur o problemă. Deși o pagină poate fi afișată perfect de programul de navigare cu care este dezvoltată, un alt program de navigare, sau chiar altă versiune a aceluiași program sau o altă rezoluție a ecranului poate fi un dezastru. XML este parțial o încercare de întoarcere la scopul originar de a specifica doar structura nu modalitatea de afișare a documentului. Totuși, XSL este oferit în plus, pentru a controla modul de afișare. Ambele formate pot avea însă utilizări eronate. Puteți conta pe asta.

XML poate fi utilizat și în alte scopuri decât acela de a descrie pagini de Web. O utilizare din ce în ce mai frecventă este aceea de limbaj de comunicare între aplicații. În particular, **SOAP (Simple Object Access Protocol –** Protocol simplu pentru accesul la obiecte) este o modalitate de a executa apeluri de tip RPC între aplicații într-un mod independent de limbaj și de sistem. Clientul construiește cererea ca mesaj XML și o transmite serverului, utilizând protocolul HTTP (descris mai departe). Serverul trimite înapoi un răspuns sub formă de mesaj XML. În acest mod pot comunica aplicații de pe sisteme heterogene.

XHTML – eXtended HyperText Markup Language

Limbajul HTML continuă să evolueze pentru a se conforma noilor cereri. Multe persoane din acest domeniu cred că în viitor majoritatea dispozitivelor cu acces la Web nu vor fi calculatoarele personale, ci dispozitive cu conexiuni fără fir, de tip PDA. Aceste dispozitive au memorie limitată pentru programe de navigare mari cu metode euristice ce încearcă să trateze într-un anumit mod paginile de Web incorecte din punct de vedere sintactic. Astfel, următorul pas după HTML 4 este un limbaj care este Foarte Selectiv. Acest limbaj este numit **XHTML (eXtended HyperText Markup Language, rom.: Limbaj extins de marcaje hipertext)** mai degrabă decât HTML 5, pentru că, de

fapt, este HTML 4 reformulat în XML. Prin aceasta vrem să spunem că marcaje precum <h1> nu au nici o însemnătate prin ele însele. Pentru a obține efectul din HTML 4 este nevoie de o definiție în fișierul XSL. XHTML este noul standard pentru Web și ar trebui folosit pentru toate paginile de Web noi pentru a asigura un maxim de portabilitate pe diverse platforme și programe de navigare.

Există șase diferențe majore și mai multe diferențe minore între XHTML și HTML 4. Să trecem acum în revistă diferențele majore. Mai întâi, paginile XHTML și programele de navigare trebuie să se supună în mod strict standardului. Gata cu paginile de Web de proastă calitate. Această proprietate a fost moștenită din XML.

În al doilea rând, toate marcajele și atributele trebuie să fie scrise cu litere mici. Marcaje ca <HTML> nu sunt valide în XHTML. Folosirea marcajelor precum <html> este acum obligatorie. Similar, <img SRC=”pic001.jpg”> este interzis pentru că are în componență un atribut scris cu litere mari.

În al treilea rând, sunt necesare marcaje de terminare, chiar și pentru </p>. Pentru marcaje care nu au un marcaj natural de terminare, cum ar fi <br>, <hr> și <img>, un caracter / trebuie să preceadă caracterul de terminare „>”, de exemplu

```
<img src=”pic001.jpg” />
```

În al patrulea rând, atributele trebuie să fie conținute între ghilimele. De exemplu,

```
<img SRC=”pic001.jpg” height=500 />
```

nu mai este permis. Valoarea 500 trebuie pusă între ghilimele, cum este numele fișierului JPEG, chiar dacă 500 este doar un număr.

În al cincilea rând, marcajele trebuie să se conțină unul pe altul într-un mod corespunzător. În trecut, acest lucru nu era necesar atât timp cât starea finală atinsă era corectă. De exemplu,

```
<center> <b> Vacation Pictures </center> </b>
```

era legal. În XHTML nu mai este. Marcajele trebuie închise în ordinea inversă în care au fost deschise.

În al șaselea rând, fiecare document trebuie să-și specifice tipul documentului. De exemplu, am văzut acest lucru în fig. 7-32. Pentru o discuție asupra schimbărilor, fie ele majore sau minore, vezi *www.w3.org*.

7.3.3 Documente Web dinamice

Până acum, modelul pe care l-am folosit este cel din Fig. 6-6: clientul transmite numele fișierului către server, care apoi întoarce fișierul. La începutul Web-ului, tot conținutul era de fapt static în acest mod (doar fișiere). Totuși, în ultimii ani, din ce în ce mai mult conținut a devenit dinamic, adică generat la cerere și nu doar stocat pe disc. Generarea de conținut poate avea loc fie la server, fie la client. Să examinăm acum pe rând fiecare din aceste cazuri.

Generare dinamică de pagini de Web la server

Pentru a vedea de ce este necesară generarea de conținut la server, să luăm în considerare utilizarea formularelor, așa cum a fost descrisă mai devreme. Atunci când un utilizator completează un formular și apasă butonul *submit*, se transmite un mesaj către server, mesaj ce arată că are în interior conținutul unui formular, împreună cu acele câmpuri completate de utilizator. Acest mesaj nu este numele unui fișier ce trebuie întors. În schimb, acest mesaj trebuie să fie oferit unui program, sau

script, pentru a fi procesat. De obicei, procesarea implică folosirea informațiilor oferite de utilizator pentru căutarea unei înregistrări într-o bază de date de pe discul serverului și generarea unei pagini HTML personalizate pentru a fi trimisă înapoi clientului. De exemplu, într-o aplicație de comerț electronic, atunci când utilizatorul face un clic pe *MERGI LA CASĂ*, programul de navigare întoarce cookie-ul ce conține produsele din coșul de cumpărături, dar la server trebuie apelat un program, sau script, care procesează acest cookie și generează o pagină HTML ca răspuns. Pagina HTML ar putea afișa un formular ce conține lista de produse din coș și ultima adresă de expediere cunoscută a utilizatorului, împreună cu o cerere de verificare a informațiilor și de specificare a modalității de plată. Etapele necesare pentru procesarea informației dintr-un formular HTML sunt ilustrate în fig. 7-33.

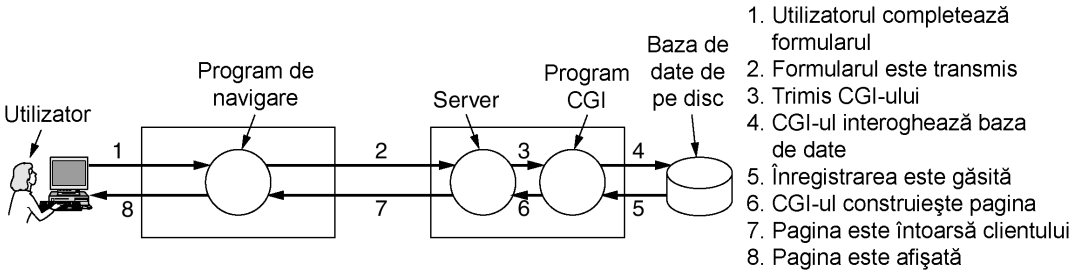


Fig. 7-33. Etapele de procesare a informației dintr-un formular HTML

Modalitatea tradițională de a trata formularele și alte pagini de Web interactive este sistemul numit **CGI** (**Common Gateway Interface –** Interfață comună de conversie). Aceasta este o interfață standardizată ce permite serverelor de Web să discute cu programele din fundal și cu scripturile care acceptă o intrare (de ex.: formulare) și să genereze pagini HTML ca răspuns. De obicei, aceste programe de fundal sunt scripturi scrise în limbajul Perl deoarece scripturile Perl sunt mai ușor și mai rapid de scris decât programele (cel puțin dacă știi să programezi în Perl). În mod obișnuit, ele sunt localizate într-un director numit *cgi-bin*, care este vizibil în URL. Câteodată un alt limbaj de scripturi, Python, este utilizat în loc de Perl.

Ca un exemplu de cât de frecvent lucrează CGI, să considerăm cazul unui produs al companiei Truly Great Products Company (*rom. Compania Produselor cu Adevărat Bune*) care vine cu o fișă de înregistrare a produsului pentru garanție. În loc de a completa această fișă, clientului i se spune să meargă la *www.tgpc.com* pentru a se înregistra on-line. Pe acea pagină există o hiper-legătură care spune

Apăsați aici pentru a va înregistra produsul

Această legătura indică spre un script Perl, să spunem *www.tgpc.com/cgi-bin/reg.perl.* Când acest script este executat fără parametri, transmite înapoi o pagină HTML conținând formularul de înregistrare. Atunci când utilizatorul completează formularul și face un clic pe *submit* se transmite un mesaj acestui script, mesaj ce conține valorile completate după modelul din fig. 7-30. Scriptul Perl analizează parametrii, adaugă o înregistrare în baza de date pentru noul client și transmite înapoi o pagină HTML ce conține numărul de înregistrare și un număr de telefon de la serviciul de asistență. Aceasta nu este singura modalitate de a trata formularele, dar este o modalitate des întâlnită. Există un număr mare de cărți despre scrierea scripturilor CGI și programarea în Perl. Câteva exemple sunt: (Hanegan, 2001; Lash, 2002; și Meltzer și Michalski, 2001).

Scripturile CGI nu sunt singura modalitate de a genera conținut dinamic la server. O altă modalitate des întâlnită este de a îngloba mici scripturi în paginile HTML și a lăsa serverul să le execute pentru a genera pagina. Un limbaj popular pentru scrierea acestor scripturi este **PHP** (**PHP: Hypertext Procesor; rom.: Procesor Hipertext**). Pentru a fi folosit, serverul trebuie să înțeleagă PHP (exact cum programul de navigare trebuie să înțeleagă XML pentru a interpreta paginile de Web scrise în XML). De obicei, serverele se așteaptă ca paginile de Web ce conțin PHP să aibă extensia *php* mai degrabă decât *html* sau *htm*.

Un mic script PHP este ilustrat în fig. 7-34; ar trebui să funcționeze pe orice server care are PHP instalat. Conține HTML normal cu excepția scriptului PHP dintre marcajele <?php ... ?>. Ceea ce generează este o pagină de Web ce afișează ceea ce știe despre programul de navigare care o apelează. Programele de navigare trimit de obicei o serie de informații odată cu cererea lor (și orice cookie aplicabil) și această informație este pusă în variabila *HTTP_USER_AGENT*. Când acest program este pus în fișierul *test.php* în directorul de WWW al companiei ABCD, tastând URL-ul *www.abcd.com/test.php* se va afișa o pagină de Web care spune utilizatorului ce program de navigare, ce limbă și ce sistem de operare folosește.

```
<html>
<body>
<h2> This is what I know about you </h2>
<?php echo $HTTP_USER_AGENT ?>
</body>
</html>
```

Fig. 7-34. O pagină HTML cu PHP înglobat

PHP este folositor în special la tratarea formularelor și este mai simplu de utilizat decât scripturile CGI. Ca un exemplu al modului său de funcționare, să considerăm exemplul din fig. 7-35(a). Această figură conține o pagină HTML normală cu un formular în interior. Singurul lucru neobișnuit la această pagină este prima linie, care spune că fișierul *action.php* va fi invocat pentru a trata parametrii după ce utilizatorul a completat și transmis formularul. Pagina afișează două căsuțe de text, una cerând numele și cealaltă vârsta. După ce aceste căsuțe au fost completate și formularul transmis, serverul analizează șirul de caractere de forma celui din fig. 7-30, punând numele în variabila *name* și vârsta în variabila *age*. Începe apoi să proceseze fișierul *action.php*, arătat în fig. 7-35(b) pentru obținerea răspunsului. În timpul procesării acestui fișier sunt executate comenzile PHP. Dacă utilizatorul a completat valorile „Barbara” și „24” în căsuțele formularului, fișierul transmis înapoi este cel dat în fig. 7-35(c). Astfel, tratarea formularelor devine extrem de simplă în PHP.

Deși PHP este ușor de utilizat, este de fapt un limbaj de programare puternic, orientat pe interfațarea dintre Web și o bază de date a serverului. Suportă variabile, șiruri de caractere, vectori și majoritatea structurilor de control din C, dar un sistem de I/E mult mai puternic decât *printf*. PHP este un program public (open source) și disponibil gratuit. A fost conceput special să lucreze bine cu Apache, care este de asemenea gratuit și care este cel mai larg utilizat server de Web din lume. Pentru mai multe informații despre PHP, vezi (Valade, 2002).

Am văzut până acum două moduri diferite de a genera pagini HTML dinamice: script-urile CGI și PHP-ul înglobat. Există și o a treia tehnică, numită **JSP** (**JavaServer Pages**), care este similară cu PHP, cu excepția faptului că partea dinamică este scrisă în limbajul de programare Java în loc de

PHP. Paginile ce folosesc această tehnică au în numele fișierului extensia *jsp*. O a patra tehnică, **ASP (Active Server Pages)**, este versiunea de la Microsoft a PHP și JavaServer Pages. Pentru generarea conținutului dinamic folosește limbajul de script proprietar al Microsoft-ului, Visual Basic Script. Paginile ce folosesc această tehnică au extensia *asp*. Alegerea dintre *PHP*, *JSP*, și *ASP* are în general mai mult de-a face cu politici (open-souce vs. Sun vs. Microsoft) decât cu tehnologia, cele trei limbaje fiind comparabile. Colecția de tehnologii pentru generarea din zbor a conținutului este uneori denumită **HTML dinamic**.

Generare dinamică de pagini de Web la client

Scripturile CGI, PHP, JSP și ASP rezolvă problema formularelor și a interacțiunilor cu bazele de date din server. Toate pot să accepte informații care vin din formulare, să caute informații într-una sau mai multe baze de date și să genereze pagini HTML cu rezultate. Ceea ce nu poate face nici unul dintre scripturi este să răspundă la mișcările mouse-ului sau să interacționeze direct cu utilizatorii. În acest scop, este necesar ca scripturile să fie înglobate în paginile HTML care sunt executate mai degrabă pe mașina clientului, decât pe mașina serverului. Începând cu HTML 4.0, astfel de scripturi erau permise folosind marcajul <script>. Cel mai popular limbaj de script la client este **JavaScript**, așa că o să aruncăm o scurtă privire asupra lui.

```
<html>
<body>
<form action="action.php" method="post">
<p> Introduceti numele: <input type="text" name="name"> </p>
<p> Introduceti varsta: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Raspuns: </h1>
Hello <?php echo $name; ?>.
Prezicere: anul urmator veti avea <?php echo $age + 1; ?> ani.
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Raspuns: </h1>
Buna, Barbara.
Prezicere: anul urmator veti avea 25 ani.
</body>
</html>
```

(c)

Fig. 7-35. (a) O pagină Web ce conține un formular.
(b) Un script PHP pentru afișarea dinamică a form-ului.
(c) Ieșirea scriptului PHP când datele de intrare sunt „Barbara” și respectiv 24.

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
 var person = test_form.name.value;
 var years = eval(test_form.age.value) + 1;
 document.open();
 document.writeln("<html> <body>");
 document.writeln("Hello " + person + ".<br>");
 document.writeln("Prezicere: la anul vei avea " + years + ".");
 document.writeln("</body> </html>");
 document.close();
}
</script>
</head>

<body>
<form>
Introduceti numele: <input type="text" name="name">
<p>
Introduceti varsta: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Fig. 7-36. Folosirea JavaScript Pentru procesarea unui formular.

JavaScript este un limbaj de script, *foarte* inspirat din câteva idei ale limbajului de programare Java. Nu este cu siguranță Java. Ca alte limbaje de script, el este un limbaj de nivel foarte înalt. De exemplu, într-o singură linie din JavaScript este posibil să se creeze o fereastră de dialog, să se aștepte introducerea de text și să se memoreze șirul rezultat într-o variabilă. Astfel de caracteristici de nivel înalt fac din JavaScript un limbaj ideal pentru crearea de pagini Web interactive. Pe de altă parte, faptul că nu este standardizat și că se modifică mai repede ca o muscă prinsă într-o mașină cu raze X, fac extrem de dificilă scrierea de programe JavaScript care să funcționeze pe toate platformele, dar poate într-o zi se va stabiliza.

Un exemplu de program în JavaScript, este cel din fig. 7-36. Ca și în fig. 7-35(a), apare un formular în care se cer numele și vârsta și care calculează vârsta persoanei în anul următor. Corpul este aproape la fel ca în exemplul PHP, principala diferență fiind declararea butonului de trimitere a datelor și asocierea unei funcții cu acest buton. Această funcție spune programului de navigare să invoce scriptul *response* la o apăsare de buton și să-i trimită formularul ca parametru.

Complet nouă aici este declararea funcției JavaScript *response* în antetul fișierului HTML, o zonă în mod normal rezervată titlurilor, culorilor de fundal și așa mai departe. Această funcție extrage valoarea câmpului *name* din formular și o păstrează ca șir în variabila *person*. De asemenea extrage valoarea câmpului *age*, o convertește la un întreg prin folosirea funcției *eval*, o incrementează cu 1 și reține rezultatul în *years*. Apoi deschide un document pentru ieșire în care face trei scrieri, folosind metoda *writeln*, și închide documentul. Documentul este un fișier HTML, după cum se poate vedea din diversele marcaje HTML din el. Programul de navigare afișează apoi documentul pe ecran.

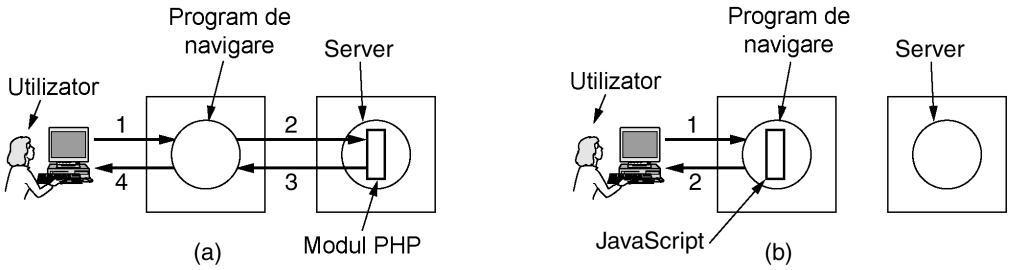


Fig. 7-37. (a) Scripting la server cu PHP. (b) Scripting la client cu JavaScript.

Este foarte important de înțeles că în timp ce fig. 7-35 și fig. 7-36 arată similar, ele sunt procesate total diferit. În fig. 7-35, după ce utilizatorul a apăsat butonul *submit*, programul de navigare strânge informația într-un șir lung, în stilul celui din fig. 7-30 și îl trimite serverului care a trimis pagina. Serverul vede numele fișierului PHP și îl execută. Scriptul PHP produce o nouă pagină HTML și acea pagină este trimisă înapoi programului de navigare pentru afișare. Cu fig. 7-36, când butonul *submit* este apăsat, programul de navigare interpretează o funcție JavaScript conținută pe pagină. Totul este făcut local, în programul de navigare. Nu se face nici un contact cu serverul. Ca o consecință, rezultatul este tipărit teoretic instantaneu, în timp ce cu PHP, poate exista o întârziere de câteva secunde înainte ca HTML-ul rezultat să ajungă la client. Diferența între utilizarea scripturilor la server și utilizarea acestora la client este ilustrată în fig. 7-37, inclusiv pașii implicați. În ambele cazuri, pașii numerotați încep după afișarea formularului. Pasul 1 constă din acceptarea datelor de intrare ale utilizatorului. Apoi urmează procesarea acestora, care diferă în cele două cazuri.

```
<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
 function factorial(n) { if (n==0) return 1; else return n * factorial(n-1); }
 var r = eval(test_form.number.value);                    // r = argument introdus de la tastatura
 document.myform.mytext.value = "Aici sunt rezultatele.\n";
 for (var i = 1; i <= r; i++)                             // tipareste o linie de la 1 la r
 document.myform.mytext.value += (i + "!= " + factorial(i) + „\n");
}
</script>
</head>

<body>
<form name="myform">
Introduceti un numar: <input type="text" name="number">
<input type="button" value="calcul factorial" onclick="response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>
```

Fig. 7-38. Un program JavaScript pentru calculul și afișarea factorialului.

Această diferență nu înseamnă că JavaScript este mai bun ca PHP. Utilizările lor sunt complet diferite. PHP (și, prin implicație, JSP și ASP) sunt utilizate când este necesară o interacțiune cu o bază de date aflată la distanță. JavaScript este utilizat când interacțiunea se face cu utilizatorul la calculatorul clientului. Este cu siguranță posibil (și des întâlnit) să existe pagini HTML care folosesc atât PHP cât și JavaScript, deși evident nu pot face același lucru pe același buton, sau să dețină același buton.

JavaScript este un limbaj de programare matur, cu toată puterea limbajelor C și Java. Are variabile, șiruri, vectori, obiecte, funcții, și toate structurile de control obișnuite. Are, de asemenea, un număr mare de facilități specifice paginilor Web, inclusiv abilitatea de a lucra cu ferestre și cadre, setarea și obținerea de cookie-uri, lucrul cu formulare, și cu hiper-legături. Un exemplu de program JavaScript care utilizează o funcție recursivă este dat în fig. 7-38.

JavaScript poate, de asemenea, să urmărească mișcarea mouse-ului peste obiectele afișate. Multe pagini Web ce conțin JavaScript au proprietatea că atunci când mouse-ul se mișcă peste un text sau o imagine, se întâmplă ceva. Deseori, imaginea se schimbă sau apare dintr-o dată un meniu. Acest tip de comportament este ușor de programat în JavaScript și conduce la pagini de Web foarte dinamice. În fig. 7-39 este dat un exemplu.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/ast/im/kitten.jpg";
document.myurl[1] = "http://www.cs.vu.nl/ast/im/puppy.jpg";
document.myurl[2] = "http://www.cs.vu.nl/ast/im/bunny.jpg";
function pop(m) {
        var urx = "http://www.cs.vu.nl/ast/im/cat.jpg";
        popupwin = window.open(document.myurl[m], "mywind", "width=250,height=250");
}
</script>
</head>

<body>
<p> <a href="#" onMouseover="pop(0); return false;"> Kitten </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;"> Puppy </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;"> Bunny </a> </p>
</body>
</html>
```

Fig. 7-39. O pagină Web interactivă care răspunde la mișcarea mouse-ului.

JavaScript nu este singura cale de a face paginile Web foarte interactive. Altă metodă populară este bazată pe folosirea **applet**-urilor. Acestea sunt mici programe Java care au fost compilate într-un cod mașină pentru un calculator virtual numit **JVM (Java Virtual Machine** – Mașina Virtuală Java). Applet-urile pot fi incluse în paginile HTML (între <applet> și </applet>) și sunt interpretate de programe de navigare care cunosc JVM. Deoarece applet-urile nu sunt executate, ci interpretate, interpretorul Java poate să le împiedice să facă Lucruri Rele. Cel puțin în teorie. În practică, autorii de applet-uri au găsit și exploatat un șir aproape nesfârșit de erori în bibliotecile Java de I/E.

Răspunsul Microsoft la applet-urile Java de la Sun au fost paginile Web cu **controale Active-X (Active-X controls)**, care sunt programe compilate pentru o mașină Pentium și sunt executate direct

în hardware. Această proprietate le face mult mai rapide și mai flexibile decât applet-urile interpretate, pentru că pot face orice poate face un program. Când Internet Explorer vede un control Active-X într-o pagină Web, îl descarcă, îi verifică identitatea și îl execută. Totuși, descărcarea și executarea de programe străine ridică probleme de securitate, la care ne vom referi în cap. 8.

Din moment ce aproape toate programele de navigare pot să interpreteze atât programe Java cât și JavaScript, un programator care vrea să facă o pagină Web foarte interactivă va putea să aleagă între două tehnici, iar dacă nu se dorește portabilitatea pe mai multe platforme, poate să aleagă și Active-X. Ca o regulă generală, programele JavaScript sunt mai ușor de scris, applet-urile Java se execută mai rapid iar controalele Active-X cel mai rapid dintre toate. De asemenea, din moment ce toate programele de navigare implementează exact aceeași JVM, dar nu există două programe de navigare care să știe aceeași versiune de JavaScript, applet-urile Java sunt mai portabile decât programele JavaScript. Pentru mai multe detalii despre JavaScript, există multe cărți, fiecare cu multe (deseori peste 1000) pagini. Câteva exemple sunt (Easttom, 2001; Harris 2001; și McFerdries, 2001).

Înainte de a părăsi subiectul conținutului dinamic al Web-ului, să recapitulăm pe scurt ce am atins până acum. Pagini Web complete se pot genera din mers, folosind diverse script-uri pe mașina server. Odată ce sunt primite de programul de navigare, ele sunt tratate ca pagini HTML normale și sunt doar afișate. Script-urile pot fi scrise în Perl, PHP, JSP sau ASP, după cum este arătat în fig. 7-40.

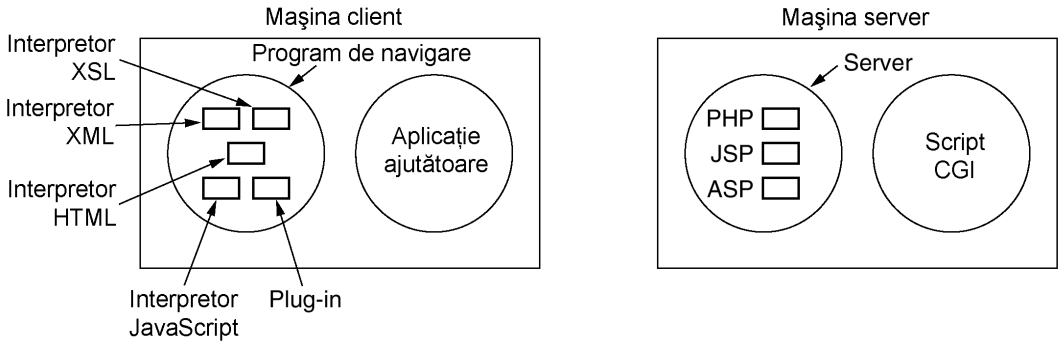


Fig. 7-40. Diverse moduri de a genera și afișa conținut.

Generarea conținutului dinamic este posibilă și în partea clientului. Paginile Web pot fi scrise în XML și convertite la HTML conform unui fișier XSL. Programele JavaScript pot să efectueze diverse calcule. În sfârșit, plug-in-urile (plug-ins) și aplicațiile ajutătoare (helper applications) pot fi folosite pentru afișarea conținutului într-o varietate de forme.

7.3.4 HTTP – HyperText Transfer Protocol

Protocolul de transfer utilizat pe Web este **HTTP (HyperText Transfer Protocol, rom.:Protocol de Transfer al Hipertextului)**. Acesta specifică ce mesaje pot trimite clienții către servere și ce răspunsuri primesc înapoi. Fiecare interacțiune constă dintr-o cerere ASCII, urmată de un răspuns MIME conform RFC 822. Toți clienții și toate serverele trebuie să se supună acestui protocol. Este definit în RFC 2616. În această secțiune vom trata câteva din proprietățile sale cele mai importante.

Conexiuni

Modul uzual prin care un program de navigare contactează un server este de a stabili o conexiune TCP pe portul 80 pe mașina serverului, deși această procedură nu este cerută formal. Avantajul de a folosi TCP este că nici programele de navigare și nici serverele nu trebuie sa se preocupe de mesajele pierdute, mesajele duplicate, mesajele lungi, sau mesajele de confirmare. Toate aceste aspecte sunt tratate de implementarea TCP.

În HTTP 1.0, după ce conexiunea era stabilită, o singură cerere era transmisă și un singur răspuns era primit înapoi. Apoi conexiunea TCP era eliberată. Într-o lume în care pagina tipică Web consta în întregime din text HTML, această metodă era adecvată. În câțiva ani însă, o pagină medie Web conținea un număr mare de iconițe, imagini și alte lucruri plăcute vederii, astfel că stabilirea unei conexiuni TCP pentru a prelua o singură iconiță a devenit un mod foarte costisitor de a opera.

Această observație a dus la apariția HTTP 1.1, care suportă **conexiuni persistente**. Cu ele, este posibilă stabilirea unei conexiuni TCP, trimiterea unei cereri și obținerea unui răspuns, apoi trimiterea unor cereri adiționale și obținerea de răspunsuri adiționale. Prin distribuirea pornirii și eliberării unei conexiuni TCP peste mai multe cereri, supraîncărcarea relativă datorată TCP-ului este mult mai mică pe fiecare cerere. Este de asemenea posibilă trimiterea cererilor prin mecanismul pipeline, adică trimiterea cererii 2 înainte ca răspunsul la cererea 1 să fi sosit.

Metode

Cu toate că HTTP a fost proiectat pentru utilizarea în Web, el a fost creat intenționat mai general decât era necesar în perspectiva aplicațiilor orientate pe obiecte. Pentru aceasta sunt suportate operațiile, denumite **metode**, care fac mai mult decât cele care doar cer o pagină Web. Această considerație generală a permis apariția SOAP. Fiecare cerere constă din una sau mai multe linii de text ASCII, în care primul cuvânt din prima linie este numele metodei cerute. Metodele incorporate sunt listate în fig. 7-41. Pentru accesarea unor obiecte generale, metode adiționale specifice obiectelor pot fi de asemenea disponibile. În numele metodelor este semnificativă utilizarea literelor mari/mici, de exemplu *GET* este o metodă acceptată, dar nu și *get*.

Metoda	Descriere
GET	Cerere de citire a unei pagini Web
HEAD	Cerere de citire a antetului unei pagini de Web
PUT	Cerere de memorare a unei pagini de Web
POST	Adăugarea la o resursă anume (de exemplu o pagină de Web)
DELETE	Ștergerea unei pagini de Web
TRACE	Tipărirea cererii care a sosit
CONNECT	Rezervat pentru o utilizare în viitor
OPTIONS	Interogarea anumitor opțiuni

Fig. 7-41. Metode de cerere standard pentru HTTP.

Metoda *GET* cere serverului să trimită pagina (prin care noi înțelegem obiect, în cel mai general caz, dar în practică de obicei doar un fișier). Pagina este codată corespunzător în MIME. Marea majoritate a cererilor către servere Web sunt metode *GET*. Forma uzuală a metodei *GET* este

```
GET fișier HTTP-1.1
```

unde *fișier* denumește resursa (fișierul) ce va fi adusă, și 1.1 este versiunea de protocol utilizat.

Metoda *HEAD* cere doar antetul mesajului, fără să ceară și pagina propriu-zisă. Această metodă poate să fie utilizată pentru a afla când s-a făcut ultima modificare, pentru a obține informații pentru indexare, sau numai pentru a verifica corectitudinea unui URL.

Metoda *PUT* este inversa metodei *GET*: în loc să citească o pagină, o scrie. Această metodă permite crearea unei colecții de pagini de Web pe un server la distanță. Corpul cererii conține pagina. Pagina poate să fie codificată utilizând MIME, caz în care liniile care urmează după *PUT* pot include *Content-Type* și antete de autentificare, pentru a demonstra că într-adevăr cel care face cererea are dreptul de a realiza operația cerută.

Similară metodei *PUT* este metoda *POST*. Și ea conține un URL, dar în loc să înlocuiască date existente, noile date se vor adăuga într-un mod generalizat. De exemplu, se poate transmite un mesaj la un grup de știri sau adăuga un fișier la un sistem de informare în rețea. În practică, nici *PUT* și nici *POST* nu sunt utilizate prea mult.

DELETE realizează ce era de așteptat: ștergerea unei pagini. Ca și la *PUT*, autentificarea și drepturile de acces joacă aici un rol important. Nu există nici o garanție asupra succesului operației *DELETE*, deoarece chiar dacă serverul dorește să execute ștergerea, fișierul poate să aibă atribute care să interzică serverului HTTP să îl modifice sau să îl șteargă.

Metoda *TRACE* este pentru verificarea corectitudinii. Ea cere serverului să trimită înapoi cererea. Această metodă este utilă când cererile nu sunt procesate corect și clientul vrea să știe ce fel de cerere a ajuns de fapt la server.

Metoda *CONNECT* nu este utilizată în prezent. Este rezervată pentru utilizări ulterioare.

Metoda *OPTIONS* asigură o modalitate pentru client de a interoga serverul despre proprietățile acestuia sau despre cele ale unui anumit fișier.

Fiecare cerere obține un răspuns ce constă din linia de stare și posibile informații suplimentare (de exemplu, o parte sau toată pagina Web). Linia de stare conține un cod de stare de trei cifre, indicând dacă cererea a fost satisfăcută și dacă nu, cauza. Prima cifră este utilizată pentru împărțirea răspunsurilor în cinci mari grupuri, ca în fig. 7-42. Codurile 1xx sunt utilizate în practică foarte rar. Codurile 2xx indică tratarea cu succes a cererii și conținutul (dacă există) este returnat. Codurile 3xx spun clientului să caute în altă parte, prin folosirea unui URL diferit, sau în propria memorie ascunsă (discutată mai târziu). Codurile 4xx indică insuccesul cererii din cauza unei erori la client, precum o cerere invalidă sau o pagină inexistentă. În fine, erorile 5xx indică o problemă în server, datorată codului său sau unei supraîncărcări temporare.

Cod	Semnificație	Exemple
1xx	Informație	100 = serverul acceptă tratarea cererii de la client
2xx	Succes	200 = cerere reușită; 204 = nu există conținut
3xx	Redirectare	301 = pagină mutată; 304 = pagina din memoria ascunsă este încă validă
4xx	Eroare la client	403 = pagină interzisă; 404 = pagina nu a fost găsită
5xx	Eroare la server	500 = eroare internă la server; 503 = încearcă mai târziu

Fig. 7-42. Grupuri de răspunsuri ale codurilor de stare.

Antete de mesaje

Linia de cerere (de exemplu linia cu metoda *GET*) poate fi urmată de linii adiționale cu mai multe informații. Acestea poartă numele de **antete de cerere**. Această informație poate fi comparată cu parametrii unui apel de procedură. Răspunsurile pot avea de asemenea **antete de răspuns**. Anumite antete pot fi folosite în orice sens. O selecție a celor mai importante este dată în fig. 7-43.

Antetul *User-Agent* permite clientului să informeze serverul asupra programului său de navigare, sistemului de operare și altor proprietăți. În fig. 7-34 am văzut că serverul avea în mod magic această informație și că o poate obține la cerere într-un script PHP. Antetul este utilizat de client pentru a-i asigura serverului această informație.

Antet	Tip	Descriere
User-Agent	Cerere	Informație asupra programului de navigare și a platformei
Accept	Cerere	Tipul de pagini pe care clientul le poate trata
Accept-Charset	Cerere	Seturile de caractere care sunt acceptabile la client
Accept-Encoding	Cerere	Codificările de pagini pe care clientul le poate trata
Accept-Language	Cerere	Limbajele naturale pe care clientul le poate trata
Host	Cerere	Numele DNS al serverului
Authorization	Cerere	O listă a drepturilor clientului
Cookie	Cerere	Trimite un cookie setat anterior înapoi la server
Date	Ambele	Data și ora la care mesajul a fost trimis
Upgrade	Ambele	Protocolul la care transmițătorul vrea să comute
Server	Răspuns	Informație despre server
Content-Encoding	Răspuns	Cum este codat conținutului (de exemplu, gzip)
Content-Language	Răspuns	Limbajul natural utilizat în pagină
Content-Length	Răspuns	Lungimea paginii în octeți
Content-Tzpe	Răspuns	Tipul MIME al paginii
Last-Modified	Răspuns	Ora și data la care pagina a fost ultima dată modificată
Location	Răspuns	O comandă pentru client pentru a trimite cererea în altă parte
Accept-Ranges	Răspuns	Serverul va accepta cereri în anumite limite de octeți
Set-Cookie	Răspuns	Serverul vrea să salveze un cookie la client

Fig. 7-43. Câteva antete de mesaje HTTP.

Cele patru antete *Accept* spun serverului ce este dispus clientul să accepte în cazul în care acesta are un repertoriu limitat despre ceea ce este acceptabil. Primul antet specifică ce tipuri MIME sunt acceptate (de exemplu, text/html). Al doilea reprezintă setul de caractere (de exemplu ISO-8859 sau Unicode-1-1). Al treilea se referă la metode de compresie (de exemplu, gzip). Al patrulea indică un limbaj natural (de exemplu, spaniola). Dacă serverul are mai multe pagini din care poate să aleagă, el poate utiliza această informație pentru a furniza clientului pagina pe care o caută. Dacă nu poate satisface cererea, este întors un cod de eroare și cererea eșuează.

Antetul *Host* denumește serverul. El este luat din URL. Antetul este obligatoriu. Este utilizat deoarece anumite adrese IP pot servi mai multe nume de DNS și serverul are nevoie de o anumită modalitate de a spune cărui calculator să-i trimită cererea.

Antetul *Authorization* este necesar pentru protecția paginilor. În acest caz, clientul trebuie să demonstreze că are dreptul de a vedea pagina cerută. Acest header este utilizat în acest scop.

Deși cookie-urile sunt tratate în RFC 2109 mai mult decât în RFC 2616, și ele au două antete. Antetul *Cookie* este utilizat de clienți pentru a întoarce serverului un cookie care a fost anterior trimis de o mașină aflată în domeniul serverului.

Antetul *Date* poate fi utilizat în ambele sensuri și conține ora și data la care a fost trimis mesajul. Antetul *Upgrade* este folosit pentru a face mai ușoară crearea unei tranziții către o viitoare (posibil incompatibilă) versiune a protocolului HTTP. Acesta permite clientului, să anunțe ce anume suportă, și serverului să afirme ceea ce folosește.

Acum am ajuns la antetele utilizate exclusiv de către server în răspunsul cererilor. Primul, *Server*, permite serverului să spună cine este și câteva proprietăți, dacă dorește.

Următoarele patru antete, toate începând cu *Content-*, permit serverului să descrie proprietățile paginii pe care o transmite.

Antetul *Last-Modified* spune când a fost modificată ultima dată pagina. Acest antet joacă un rol important în mecanismul de memorie ascunsă.

Antetul *Location* este utilizat de server pentru a informa clientul că ar trebui să utilizeze un alt URL. Acesta poate fi folosit dacă pagina a fost mutată, sau pentru a da permisiunea mai multor URL-uri de a referi aceeași pagină (posibil pe servere diferite). Este de asemenea utilizată pentru companiile care au o pagină de Web principală în domeniul *com*, dar care redirecționează clienții la o pagină națională sau regională în funcție de adresa lor IP sau limba preferată.

Dacă o pagină este foarte mare, un client mic poate nu o dorește dintr-o dată. Unele servere acceptă cereri în anumite intervale de octeți, astfel că pagina poate fi citită în mai multe unități mai mici. Antetul *Accept-Ranges* anunță asentimentul severului de a trata acest tip de cerere parțială de pagini.

Al doilea antet pentru cookie, *Set-Cookie*, se referă la modul în care serverele trimit cookie-uri la clienți. Este de așteptat salvarea cookie-ului de către client și returnarea acestuia la cereri ulterioare ale serverului.

Exemplu de utilizare HTTP

Deoarece HTTP este un protocol ASCII, este destul de ușor pentru o persoana aflată la un terminal (ca opus al programului de navigare) să vorbească direct cu serverele de Web. Este necesară doar o conexiune TCP la portul 80 pe server. Cititorii sunt încurajați să încerce personal acest scenariu (preferabil dintr-un sistem UNIX, deoarece anumite sisteme nu returnează starea conexiunii).

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>IETF RFC Page</title>
<script language="javascript">
function url() {
var x = document.form1.number.value
if (x.length == 1) { x = "000" + x }
if (x.length == 2) { x = "00" + x }
if (x.length == 3) { x = "0" + x }
document.form1.action = „/rfc/rfc" + x + „.txt"
document.form1.submit
}
</script>
</head>
```

Fig. 7-44. Primele linii ale fișierului *www.ietf.org/rfc.html*.

Următoarea secvență de comenzi va realiza acest lucru:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org

close
```

Această secvență de comenzi pornește o conexiune telnet (adică TCP) pe portul 80 al serverului Web al IETF, *www.ietf.org*. Rezultatul sesiunii este redirectat către fișierul *log* pentru o inspecție ulterioară. Apoi urmează comanda *GET* denumind fișierul și protocolul. Următoarea linie este antetul obligatoriu *Host*. Linia rămasă liberă este de asemenea cerută. Ea semnalează serverului că nu mai sunt antete de cerere. Comanda *close* indică programului de telnet să întrerupă conexiunea.

Fișierul *log* poate fi inspectat folosind un editor. Acesta ar trebui să înceapă similar cu liniile de cod din fig. 7-44, cu excepția unei modificări recente de către IETF.

Primele trei linii reprezintă ieșirea programului telnet, și nu de la serverul la distanță. Linia ce începe cu HTTP/1.1 este răspunsul IETF prin care spune că este dispus să vorbească HTTP/1.1 cu tine. Apoi urmează un număr de antete și apoi conținutul. Am văzut deja toate antetele, cu excepția lui *ETag* care este un identificator de pagină unic, referitor la memoria ascunsă, și *X-Pad* care nu este standardizat, probabil o cale de scăpare pentru vreun program de navigare cu erori.

7.3.5 Îmbunătățiri ale performanței

Popularitatea Web-ului aproape că a fost propria sa distrugere. Servere, rutere și linii folosite de Web sunt adesea supraîncărcate. Multă lume a început să denumească WWW-ul ca World Wide Wait (rom: așteptare de întindere planetară). Ca o consecință a acestor întârzieri fără sfârșit, cercetătorii au dezvoltat diverse tehnici pentru îmbunătățirea performanțelor. Vom examina acum trei dintre ele: memoria ascunsă, replicarea serverelor și rețele de livrare a conținutului.

Memoria ascunsă

Un mod simplu de a îmbunătăți performanța este de a salva paginile care au fost cerute pentru cazul în care ele vor fi utilizate din nou. Această tehnică este efectivă în special pentru paginile care sunt vizitate foarte mult, ca *www.yahoo.com* și *www.cnn.com*. Paginile pot fi păstrate pentru utilizări ulterioare în **memoria ascunsă (eng.: cache)**. Procedura uzuală este ca un proces, denumit **proxy (rom.: delegat)**, să întrețină această memorie. Pentru a utiliza memoria ascunsă, un program de navigare poate fi configurat să adreseze toate cererile de pagini proxy-ului, și nu serverului real unde se află pagina respectivă. Dacă proxy-ul are pagina, o returnează imediat. Dacă nu, aduce pagina de la server, o adaugă în memoria ascunsă pentru utilizarea ulterioară și o întoarce clientului care a cerut-o.

Două întrebări importante aferente memoriei ascunse sunt:

1. Cine ar trebui să dețină memoria ascunsă?
2. Cât de mult timp ar trebui să stea paginile în memoria ascunsă?

Există mai multe răspunsuri la prima întrebare. PC-urile individuale de obicei rulează proxy-uri, deci pot să caute rapid pagini vizitate anterior. Într-un LAN al unei companii, proxy-ul este în general o mașină ce poate fi accesată de toate mașinile din acel LAN, astfel că dacă un utilizator se uită la o anumită pagină și apoi alt utilizator din același LAN vrea aceeași pagină, ea poate fi adusă din memoria ascunsă a proxy-ului. Multe ISP-uri rulează proxy-uri, pentru a accelera accesul clienților

lor. De obicei toate memoriile ascunse operează în același timp, deci cererile se duc inițial la proxy-ul local. Dacă eșuează, proxy-ul local cere pagina proxy-ului din LAN. Dacă și aceasta eșuează, proxy-ul din LAN încearcă la proxy-ul ISP-ului. Ultimul trebuie să reușească să aducă paginile, fie din memoria sa ascunsă, fie de la o memorie ascunsă de nivel superior, fie de la însuși serverul ce deține pagina. O schemă ce include mai multe memorii ascunse care pot fi încercate în secvență este denumită **memorie ascunsă ierarhică (hierarchical caching)**. O posibilă implementare este ilustrată în fig. 7-45.

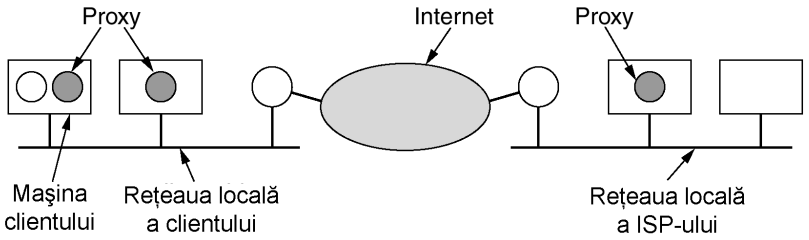


Fig. 7-45. Memorie ascunsă ierarhică cu 3 proxy-uri .

Cât timp ar trebui paginile să rămână în memoria ascunsă este un pic mai dificil de aflat. Anumite pagini nu ar trebui să fie păstrate deloc în memoria ascunsă. De exemplu, o pagină ce conține prețurile pentru cele mai active 50 de acțiuni la bursă se schimbă la fiecare secundă. Dacă s-ar păstra în memoria ascunsă, un utilizator ce ia o astfel de copie ar lua date **vechi** (adică depășite). Pe de altă parte, din momentul în care schimbul de acțiuni s-a închis pe ziua respectivă, pagina va rămâne validă ore sau zile, până când începe următoarea licitație. Astfel, eficiența menținerii unei pagini în memoria ascunsă poate varia foarte mult în timp.

Problema-cheie în determinarea eliminării unei pagini din memoria ascunsă este legată de vechimea pe care utilizatorii sunt dispuși să o accepte (din moment ce paginile din memoria ascunsă sunt ținute pe disc, cantitatea de memorare consumată reprezintă rareori o problemă). Dacă un proxy elimină repede paginile, el va întoarce rar o pagină veche, dar nu va fi prea eficient (adică va avea o rată scăzută de succes). Dacă păstrează paginile prea mult, poate avea o rată mai mare dar cu prețul de a întoarce deseori pagini cu informație expirată.

Există două abordări în tratarea acestei probleme. Prima utilizează o euristică pentru a ști cât timp să mențină fiecare pagină. O euristică des întâlnită este cea în care se ține cont de antetul *Last-Modified* (vezi fig. 7-43). Dacă o pagină a fost modificată cu o oră în urmă, se ține în memoria ascunsă o oră. Dacă a fost modificată cu un an în urmă, este evident o pagină foarte veche (de exemplu, o listă a zeilor din mitologia greacă și cea romană), deci poate fi păstrată în memoria ascunsă pentru un an, cu o probabilitate rezonabilă că nu se va modifica în decursul anului. Deși această euristică funcționează bine în practică, ea întoarce, totuși, pagini vechi din când în când.

Cealaltă abordare este mai costisitoare dar elimină posibilitatea paginilor vechi prin utilizarea unor caracteristici speciale ale RFC 2616 care tratează administrarea memoriei ascunse. Una din cele mai utilizate caracteristici este antetul de cerere *If-Modified-Since*, pe care un proxy poate să-l trimită unui server. El specifică pagina pe care o vrea proxy-ul și momentul la care pagina din memoria ascunsă a fost modificată ultima dată (din antetul *Last-Modified*). Dacă pagina nu a mai fost modificată de atunci, serverul trimite înapoi un scurt mesaj *Not Modified* (codul de stare 304 din fig. 7-42), care indică proxy-ului să utilizeze pagina din memoria ascunsă. Dacă pagina a mai fost modificată de atunci, este returnată noua pagină. În timp ce pentru această abordare este întotdeauna ne-

voie de un mesaj de cerere și unul de răspuns, mesajul de răspuns va fi foarte scurt când intrarea în memoria ascunsă este încă validă.

Aceste două abordări pot fi combinate ușor. Pentru primul ΔT după aducerea paginii, proxy-ul doar returnează pagina clienților ce o cer. După ce pagina a stat un timp în memoria ascunsă, proxy-ul utilizează mesajul *If-Modified-Since* pentru a verifica valabilitatea acesteia. Alegerea ΔT implică invariabil o euristică, depinzând de cât de mult timp a trecut de la modificarea paginii.

Paginile Web cu conținut dinamic (de exemplu, cele generate de un script PHP) nu ar trebui niciodată păstrate în memoria ascunsă, deoarece parametrii pot fi diferiți la următoarea accesare. Pentru a trata aceasta și alte cauze, există un mecanism general prin care un server instruiește toate proxy-urile până la client să nu folosească din nou pagina curentă până nu îi verifică valabilitatea. Acest mecanism poate fi folosit de asemenea pentru orice pagină pasibilă să fie modificată curând. Diverse mecanisme de control al memoriei ascunse sunt definite în RFC 2616.

Mai există o abordare în îmbunătățirea performanței, memoria ascunsă proactivă. Când un proxy aduce o pagină de la un server, el poate inspecta pagina pentru a vedea dacă ea conține hiperlegături. Dacă este așa, poate să lanseze cereri serverelor corespunzătoare pentru a preîncărca în memoria ascunsă paginile la care punctează, pentru cazul în care va fi nevoie de ele. Această tehnică poate reduce timpul de acces pentru cererile ulterioare, dar poate, la fel de bine, să inunde liniile de comunicație cu pagini care nu vor fi niciodată necesare.

În mod clar, memoria ascunsă în Web este departe de a fi banală. Mult mai multe se pot spune despre ea. De fapt, s-au scris cărți întregi pe această temă, de exemplu (Rabinovich și Spatscheck, 2002; și Wessels, 2001). Dar este timpul ca noi să trecem la un alt subiect.

Replicarea serverelor

Memoria ascunsă este o tehnică orientată spre client pentru îmbunătățirea performanțelor, dar există și tehnici orientate pe server. Cea mai întâlnită abordare pentru îmbunătățirea performanțelor serverelor este replicarea conținutului lor în mai multe locuri separate. Această tehnică este câteodată denumită **oglindire (mirroring)**.

Într-o utilizare tipică a oglindirii într-o companie, pagina principală de Web conține câteva imagini cu legături către siturile Web regionale, de exemplu siturile din est, vest, nord și sud. Utilizatorul urmează legătura cea mai apropiată. Din acel moment, toate cererile se duc la serverul selectat.

Siturile oglindite sunt în general complet statice. Compania decide unde să plaseze copiile, dispune de un server în fiecare regiune, și plasează (mai mult sau mai puțin) întregul conținut în fiecare loc (omițând, probabil, dezăpezitoarele în situl de la Miami și șezlongurile în situl din Anchorage). Alegerea siturilor rămâne în general stabilă luni sau chiar ani de zile.

Din păcate, Web-ul prezintă un fenomen cunoscut ca **aglomerare bruscă (flash crowds)** în care un sit Web care era anterior necunoscut, nevizitat, aproape mort, devine dintr-o dată centrul universului. De exemplu, până pe 6 noiembrie 2000, situl Web al secretarului de stat din Florida, *www.dos.state.fl.us*, informa tacit despre întâlnirile cabinetului statului Florida și oferea instrucțiuni pentru a deveni notar în Florida. Dar pe 7 noiembrie 2000, când președinția Statelor Unite depindea dintr-o dată de câteva mii de voturi disputate în câteva provincii din Florida, a devenit unul din primele 5 situri Web din lume. Evident, nu a putut suporta încărcarea și aproape că a murit strivit sub ea.

Este necesar un mod prin care un sit Web, ce observă dintr-o dată o cerere masivă a traficului, să se cloneze automat în atâtea locații cât este necesar și să păstreze aceste situri operaționale până când trece furtuna, moment în care oprește majoritatea sau chiar totalitatea acestora. Pentru a avea

această abilitate, un sit are nevoie de o înțelegere prealabilă cu o companie care deține mai multe situri, prin care se pot crea replici la cerere și pentru care se plătește în funcție de capacitatea pe care o folosește în realitate.

O strategie și mai flexibilă este de a crea replici dinamice la nivel de pagini, în funcție de unde vine traficul. Câteva cercetări pe această temă sunt raportate în (Pierre ș.a.., 2001; și Pierre ș.a.., 2002).

Rețele de livrare de conținut

Culmea capitalismului este că cineva a descoperit cum să câștige bani din World Wide Wait. Funcționează în felul următor. Companiile denumite **CDN (Content Delivery Networks, rom: rețele de livrare de conținut)** vorbesc cu deținătorii conținutului (situri muzicale, ziare, și alții care doresc să facă disponibil conținutul ușor și rapid) și se oferă să livreze acest conținut utilizatorilor finali în mod eficient, contra cost. După semnarea contractului, deținătorul conținutului oferă CDN-ului conținutul sitului său Web pentru preprocesare (care va fi discutată imediat) și apoi distribuție.

Apoi CDN vorbește cu un număr mare de ISP-uri și se oferă să îi plătească bine pentru dreptul de a plasa un server administrat la distanță, plin de conținut valoros, pe LAN-urile lor. Nu numai că este o sursă de venit, dar asigură de asemenea clienților ISP-urilor timpi de răspuns excelenți pentru a ajunge la conținutul CDN-ului, oferind astfel ISP-ului un avantaj competitiv față de alte ISP-uri care nu au acceptat oferta CDN-ului. În aceste condiții, colaborarea cu un CDN este ceva la mintea cocoșului pentru ISP. Ca o consecință, cele mai mari CDN-uri au mai mult de 10.000 de servere răspândite în toată lumea.

```
<html>
<head><title>Furry Video</title></head>
<body>
<h1>Furry Video's Product List</h1>
<p>Click below for free samples.</p>

<a href="bears.mpg">Bears Today</a><br>,
<a href="bunnies.mpg">Funny Bunnies</a><br>
<a href="mice.mpg">Nice Mice</a><br>
</body>
</html>
```

(a)

```
<html>
<head><title>Furry Video</title></head>
<body>
<h1>Furry Video's Product List</h1>
<p>Click below for free samples.</p>

<a href="http://cdn-server.com/furryvideo/bears.mpg">Bears Today</a><br>,
<a href="http://cdn-server.com/furryvideo/bunnies.mpg">Funny Bunnies</a><br>
<a href="http://cdn-server.com/furryvideo/mice.mpg">Nice Mice</a><br>
</body>
</html>
```

(b)

Fig. 7-46. (a) Pagina Web originală. (b) Aceeași pagină după transformare.

Cu un conținut replicat pe mii de situri în lumea întreagă, există în mod clar un potențial ridicat pentru îmbunătățirea performanțelor. Cu toate acestea, pentru o funcționare bună, trebuie să existe o modalitate prin care să se redirecteze cererea clientului la cel mai apropiat server CDN, preferabil unul aflat la ISP-ul clientului. De asemenea, această redirectare trebuie făcută fără modificarea DNS-ului sau a oricărei alte părți a infrastructurii standard a Internet-ului. O descriere puțin simplificată despre cum lucrează Akamai, cel mai mare CDN, este oferită în continuare.

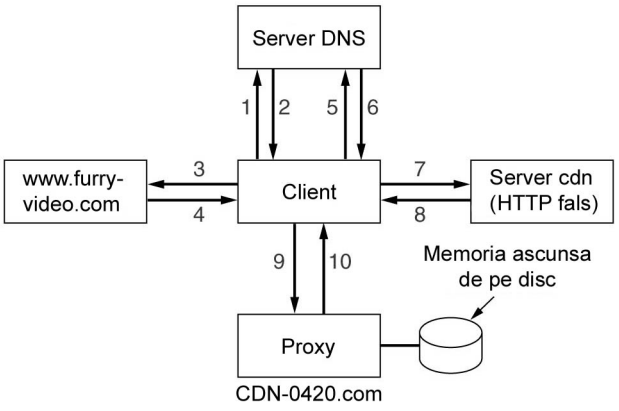
Întregul proces începe din momentul în care furnizorul conținutului trimite CDN-ului situl său Web. Apoi CDN-ul trece fiecare pagină printr-un preprocesor care înlocuiește toate URL-urile cu unele modificate. Modelul de lucru din spatele acestei strategii este acela că situl Web al furnizorului de conținut este constituit din multe pagini foarte mici (doar text HTML), dar că aceste pagini au de obicei referințe către fișiere mari, precum imagini, audio și video. Paginile HTML modificate sunt păstrate pe serverul furnizorului de conținut și sunt aduse în mod obișnuit; doar imaginile, comunicațiile audio și video merg pe serverele CDN-ului.

Pentru a vedea cum funcționează această schemă în realitate, se consideră pagina Web a lui Furry (rom.: îmblănit) Video din fig. 7-46(a). După preprocesare, ea este transformată în fig. 7-46(b) și plasată pe serverul Furry Video ca *www.furryvideo.com/index.html*.

Când un utilizator introduce ca URL *www.furryvideo.com*, DNS-ul întoarce adresa IP a sitului Furry Video, permițând ca pagina (HTML) principală să fie adusă în mod obișnuit. Când se face clic pe oricare din hiper-legături, programul de navigare cere DNS-ului să caute *cdn-server.com*, ceea ce acesta chiar face. Programul de navigare trimite apoi o cerere HTTP către această adresă IP, contând pe faptul că va primi înapoi un fișier MPEG.

Aceasta nu se întâmplă deoarece *cdn-server.com* nu găzduiește nici un conținut. În schimb, acesta este pe serverul fals de HTTP al CDN-ului. El examinează numele fișierului și numele serverului pentru a afla care pagină este necesară cărui furnizor de conținut. De asemenea, examinează adresa IP a cererii sosite și o caută în baza sa de date pentru a determina unde este posibil să se afle utilizatorul. Cu o astfel de informație, el determină care servere CDN de conținut pot oferi utilizatorului serviciul cel mai bun. Această decizie este dificilă deoarece serverul situat geografic cel mai aproape poate să nu fie cel mai apropiat în termeni de topologie de rețea, iar cel mai apropiat în termeni de topologie de rețea poate fi foarte aglomerat în acel moment. După ce se face o alegere, *cdn-server.com* trimite înapoi un răspuns cu codul de stare 301 și un antet *Location* cu URL-ul fișierului pe serverul CDN de conținut cel mai apropiat de client. Pentru acest exemplu, să presupunem că URL-ul este *www.CDN-0420.com/furryvideo/bears.mpg*. Programul de navigare procesează apoi acest URL în mod normal pentru a obține fișierul MPEG real.

Pașii urmați sunt ilustrați în fig. 7-47. Primul pas este căutarea *www.furryvideo.com* pentru a obține adresa lui IP. După aceea, pagina HTML poate fi adusă și afișată în mod obișnuit. Pagina conține trei hiper-legături la *cdn-server* [vezi fig. 7-46(b)]. Când, să zicem, este selectată prima, este căutată (pasul 5) și returnată (pasul 6) adresa sa de DNS. Când o cerere pentru *bears.mpg* este trimisă la *cdn-server* (pasul 7), clientul este înștiințat că trebuie să se ducă de fapt la *CDN-0420.com* (pasul 8). Când face acest lucru (pasul 9), i se dă fișierul din memoria ascunsă a proxy-ului (pasul 10). Proprietatea care face ca întregul mecanism să funcționeze este pasul 8, serverul fals de HTTP redirectând clientul la un proxy CDN aflat aproape de client.



1. Caută www.furryvideo.com
2. Este întoarsă adresa IP a lui Furry
3. Pagina HTML este cerută de la Furry
4. Este întoarsă o pagină HTML
5. După selecția cu mouse-ul, se caută cdn-server.com
6. Este întoarsă adresa IP a lui cdn-server
7. Bears.mpg este cerut de la cdn-server
8. Clientul este redirecționat către CDN-0420
9. Este cerut fișierul bears.mpg
10. Fișierul bears.mpg este întors din memoria ascunsă.

Fig. 7-47. Pașii în căutarea unui URL când se folosește un CDN

Serverul CDN la care este redirectat clientul este în mod tipic un proxy cu o memorie ascunsă preîncărcată cu conținutul cel mai important. Dacă totuși cineva cere un fișier care nu este în memoria ascunsă, acesta este adus de la serverul real și dispus în memoria ascunsă pentru o utilizare ulterioară. Făcând din serverul de conținut un proxy și nu o replică completă, CDN are abilitatea de a schimba dimensiunea discului, timpul de preîncărcare și diverși parametri de performanță.

Mai multe despre rețele de livrare a conținutului găsiți în (Hull, 2002; și Rabinovich și Spatscheck, 2002).

7.3.6 Web-ul fără fir

Există un interes considerabil pentru dispozitivele mici, portabile, capabile să acceseze Web-ul printr-o legătură fără fir. De fapt, primii pași în această direcție au fost deja făcuți. Cu siguranță că vor fi o mulțime de schimbări în acest domeniu în anii ce vin, dar tot merită să examinăm câteva din idele actuale legate de web-ul fără fir, pentru a vedea unde suntem acum și încotro ne putem îndrepta. Ne vom concentra asupra primelor două sisteme Web fără fir de scară largă care au spart piața: WAP și i-mode.

WAP-The Wireless Application Protocol (Protocolul pentru aplicații fără fir)

O dată ce Internet-ul și telefoanele mobile au devenit lucruri comune, nu a durat mult până când cuiva i-a venit ideea să le combine într-un telefon mobil cu ecran încorporat pentru acces fără fir la poșta electronică și la Web. Acel „cineva" a fost consorțiul condus inițial de Nokia, Ericsson, Motorola și phone.com (fosta Unwired Planet) și care acum se laudă cu sute de membri. Sistemul se numește **WAP (Wireless Application Protocol** - protocolul pentru aplicații fără fir).

Un dispozitiv WAP poate fi un telefon mobil îmbunătățit, PDA, sau calculator portabil fără servicii pentru voce. Specificația le permite pe toate și multe altele. Ideea de bază este să se folosească infrastructura digitală fără fir existentă. Utilizatorii pot accesa o poartă (eng.: gateway) WAP prin legătura fără fir și îi pot trimite cereri de pagini Web. Apoi, poarta controlează memoria ascunsă pentru pagina cerută. Dacă există, o trimite; dacă nu există, o ia de pe Internet-ul cu fir. În esență, această înseamnă că WAP 1.0 este un sistem cu comutare de circuite cu o taxă de

conectare pe minut relativ mare. Pentru a scurta o poveste lungă, oamenilor nu le-a plăcut sa acceseze Internet-ul pe un ecran mic și plătind la minut, astfel că WAP-ul a fost un fel de nereușită (deși au mai fost și alte probleme). În orice caz, WAP-ul și competitorul sau, i-mode (prezentat mai jos), par să conveargă spre o aceeași tehnologie, astfel că WAP 2.0 ar mai putea să fie un mare succes. Întrucât WAP 1.0 a fost prima încercare pentru Internet-ul fără fir, merită să fie descris cel puțin pe scurt.

WAP este de fapt o stivă de protocoale pentru accesarea Web-ului, optimizată pentru conexiuni cu o bandă de transfer mică folosind dispozitive fără fir ce au un procesor lent, puțină memorie și un ecran mic. Aceste cerințe sunt evident diferite de cele pentru un PC standard de birou, scenariu care duce la niște diferențe între protocoale. Nivelurile sunt prezentate în fig. 7-48.

Mediul aplicațiilor fără fir (WAE)
Protocolul sesiune fără fir (WSP)
Protocolul tranzacție fără fir (WTP)
Securitatea la nivelul transport fără fir (WTLS)
Protocolul pentru datagrame fără fir (WDP)
Nivelul fizic (GSM, CDMA, D-AMPS, GPRS, etc.)

Fig. 7-48. Stiva de protocoale WAP.

Nivelul cel mai de jos suportă toate sistemele de telefonie mobilă existentă, inclusiv GSM, D-AMPS și CDMA. Rata de transfer pentru WAP 1.0 este de 9600 bps. Deasupra acestora se află protocolul pentru datagrame, **WDP (Wireless Datagram Protocol -** protocolul pentru datagrame fără fir), care este de fapt UDP. Apoi vine un nivel pentru securitate, evident necesar într-un sistem fără fir. WTLS este un subset al SSL-ului de la Netscape, la care ne vom uita în cap. 8. Deasupra acestuia este un nivel tranzacție sigură sau nesigură, care se ocupă de cereri și răspunsuri. Acest nivel înlocuiește TCP, care nu este folosit peste legătura prin aer din motive legate de eficiență. Apoi vine un nivel sesiune, care este similar cu HTTP/1.1, dar cu câteva restricții și extensii pentru motive de optimizare. Deasupra acestuia se află micro-programul de navigare (WAE).

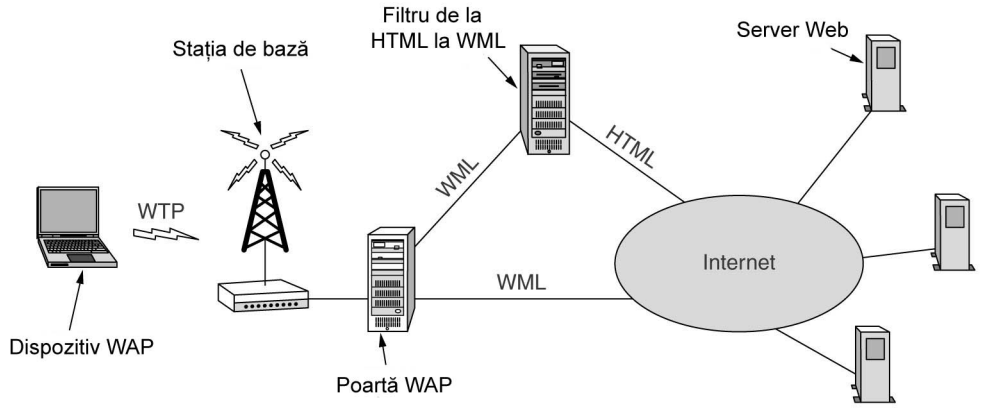


Fig. 7-49. Arhitectura WAP.

În afara costului, celălalt aspect care cu siguranță a afectat acceptarea WAP-ului este faptul că nu folosește HTML. În locul acestuia, nivelul WAE folosește un limbaj de marcare numit **WML (Wireless Markup Language** – limbajul de marcare fără fir), care este o aplicație a XML. Drept consecință, în principiu, un dispozitiv WAP nu poate accesa decât acele pagini care au fost convertite la WML. Oricum, având în vedere că asta restricționează mult valoarea WAP-ului, arhitectura reclamă un filtru direct de la HTML la WML pentru a crește mulțimea paginilor disponibile. Arhitectura este ilustrată în fig. 7-49.

Ca să fim corecți, WAP-ul a fost, probabil, puțin înaintea vremii sale. Când WAP-ul a fost lansat prima dată, XML abia era cunoscut în afara W3C și astfel presa a anunțat lansarea sa spunând **WAP NU FOLOSEȘTE HTML**. Un titlu mai clar ar fi fost: **WAP DEJA FOLOSEȘTE NOUL STANDARD HTML.** Dar cum răul fusese făcut, a fost greu de reparat și WAP 1.0 nu a prins niciodată. Vom rediscuta WAP-ul după ce ne vom uita mai întâi la competitorul său major.

I-Mode

În timp ce un consorțiu multi-industrial de companii de telecomunicații și de calculatoare a fost ocupat cu construcția unui standard deschis folosind cea mai avansată versiune de HTML disponibilă, alte dezvoltări aveau loc în Japonia. Acolo, o japoneză, Mari Matsunaga, a inventat o altă soluție pentru Web-ul fără fir numită **i-mode (information mode** – modul informație). Ea a convins divizia „fără fir" a fostului monopol de telefonie japoneză că ideea sa era corectă și în februarie 1999 NTT DoCoMo (în traducere literală: Compania Japoneză pentru Telefoane și Telegraf oriunde te-ai duce) a lansat serviciul în Japonia. În 3 ani a avut peste 35 de milioane de abonați japonezi, care puteau accesa peste 40.000 de situri Web speciale i-mode. În plus, a mai făcut ca majoritatea companiilor de telecomunicații să saliveze după succesul său financiar, mai ales datorită faptului că WAP nu părea sa ducă nicăieri. Să vedem acum ce este i-mode și cum funcționează.

Sistemul i-mode are trei componente de bază: un nou sistem de transmisie, un nou telefon și un nou limbaj pentru proiectarea paginilor Web. Sistemul de transmisie constă în două rețele separate: rețeaua de telefonie mobilă cu comutare de circuite existentă (oarecum comparabilă cu D-AMPS) și o nouă rețea cu comutare de pachete construită în mod special pentru serviciile i-mode. Modul voce folosește rețeaua cu comutare de circuite și este taxat la fiecare minut de conectare. I-mode folosește rețeaua cu comutare de pachete și este întotdeauna activ (la fel ca la ADSL sau la cablu), astfel că nu există taxarea pentru timpul de conectare. În locul acesteia, există o taxă pentru fiecare pachet trimis. Momentan nu este posibil să fie folosite ambele rețele în același timp.

Telefoanele arată ca niște telefoane mobile cărora li s-a adăugat un mic ecran. NTT DoCoMo promovează masiv dispozitivele i-mode ca fiind mai degrabă telefoane mobile decât terminale Web fără fir, deși ele chiar asta sunt. De fapt, probabil că majoritatea clienților nici nu sunt conștienți că sunt conectați la Internet. Ei consideră dispozitivele lor i-mode ca fiind telefoane mobile cu facilități sporite. Pentru a păstra acest model de i-mode la nivel de serviciu, telefoanele nu pot fi programate de utilizatori, deși ele conțin echivalentul unui PC din 1995 și ar putea probabil rula Windows 95 sau UNIX.

Când telefonul i-mode este pornit, utilizatorului îi este prezentată o listă cu categoriile de servicii aprobate oficial. Sunt mult peste 1000 de servicii grupate în aproximativ 20 de categorii. Fiecare serviciu, care este de fapt un mic sit Web i-mode, este oferit de către o companie independentă. Categoriile importante din meniul oficial includ poșta electronică, știri, meteo, sport, jocuri, cumpărături, hărți, horoscop, distracție, călătorii, ghiduri regionale, tonuri ale soneriei, rețete, jocuri de noroc, servicii bancare si cotațiile bursei. Serviciul este oarecum orientat către adolescenți și oameni de 20-

30 de ani, care au tendința să se atașeze de jucăriile electronice, mai ales dacă sunt în culori frumos asortate. Simplul fapt că peste 40 de companii vând tonuri ale soneriei spune ceva. Cea mai populară aplicație este poșta electronică, care permite mesaje de până la 500 de octeți și din acest motiv este văzută ca o mare îmbunătățire față de SMS (Short Message Service – serviciul de mesaje scurte) care permite mesaje de numai 160 de octeți. Jocurile sunt și ele populare.

Sunt de asemenea peste 40.000 de situri Web i-mode, dar ele trebuie accesate mai degrabă scriindu-se URL-ul lor, decât selectându-le dintr-un meniu. Dintr-un punct de vedere, lista oficială este ca un portal Internet care permite altor situri Web să fie accesate prin selecție în loc să li se scrie URL-ul.

NTT DoCoMo controlează îndeaproape serviciile oficiale. Pentru a fi acceptat pe listă, un serviciu trebuie să îndeplinească o serie de criterii publice. De exemplu, un serviciu nu trebuie să aibă o influență negativă asupra societății, dicționarele japonez-englez trebuie să aibă suficiente cuvinte, serviciile cu tonuri pentru sonerie trebuie să adauge frecvent noi tonuri și nici un sit nu poate să promoveze comportarea vicioasă sau să se reflecte negativ asupra NTT DoCoMo (Frangle, 2002). Cele 40.000 de situri Internet pot face orice vor ele.

Modelul afacerii i-mode este atât de diferit de acela al Internet-ului convențional încât merită explicat. Taxa pentru abonamentul de bază i-mode este de câțiva dolari pe lună. Cum există o taxă pentru fiecare pachet primit, abonamentul de bază include și un mic număr de pachete. Ca alternativă, clientul poate opta pentru un abonament cu mai multe pachete gratuite, cu o taxă pe pachet ce scade repede pe măsură ce trece de la 1 MB pe lună la 10 MB pe lună. Dacă pachetele gratuite sunt folosite până la jumătatea lunii, pot fi cumpărate on-line alte pachete adiționale.

Pentru a folosi un serviciu trebuie să te abonezi la el, fapt care se realizează printr-o simplă selecție și introducerea codului PIN personal. Majoritatea serviciilor oficiale costă în jur de 1-2 pe lună. NTT DoCoMo adaugă taxa la factura de telefon și transferă 91% celui care oferă serviciul, păstrând 9%. Dacă un serviciu neoficial are 1 milion de clienți, trebuie sa trimită 1 milion de facturi de (aproximativ) 1$ în fiecare lună. Dacă acel serviciu devine oficial, NTT DoCoMo se ocupă de taxare și transferă lunar 910.000$ în contul din bancă al serviciului. A nu avea de manipulat note de plată este un mare stimulent pentru ca cineva să devină distribuitor oficial de servicii, ceea ce generează venituri mai mari pentru NTT DoCoMo. De asemenea, fiind oficial ajungi în meniul inițial, ceea ce face situl tău mult mai ușor de găsit. Factura utilizatorului include convorbirile, taxele de abonamente i-mode, taxele de abonamente pentru servicii și pachetele suplimentare.

În ciuda succesului său masiv în Japonia, nu este de loc clar că i-mode va prinde și în SUA și Europa. Din unele puncte de vedere, situația din Japonia este diferită de aceea din Vest. În primul rând, majoritatea potențialilor clienți din Vest (spre exemplu adolescenții, studenții și oamenii de afaceri) au deja un PC cu ecran mare acasă și aproape sigur o conexiune la Internet de cel puțin 56 Kbps, adesea mult mai rapidă. În Japonia, puțini oameni au PC-uri conectate la Internet acasă, pe de o parte din cauza lipsei de spațiu, dar și din cauza taxelor exorbitante ale NTT pentru serviciile de telefonie locală (undeva în jur de 700$ pentru instalarea unei linii și 1.50$ pe oră pentru convorbiri locale). Pentru majoritatea utilizatorilor, i-mode este singura lor conexiune la Internet.

În al doilea rând, locuitorii din Vest nu sunt obișnuiți sa plătească 1$ pe lună pentru a accesa situl Web al CNN, 1$ pe lună pentru a accesa situl Yahoo, 1$ pe lună pentru a accesa situl Google și așa mai departe, fără să mai menționăm câțiva dolari pentru fiecare MB descărcat. Majoritatea distribuitorilor de Internet din Vest au acum o taxă fixă pe lună, independentă de utilizarea reală, în mare măsură ca răspuns la cererea clienților.

În al treilea rând, pentru mulți japonezi, perioada de vârf în care folosesc i-mode este perioada în care se deplasează la sau de la serviciu sau școală în tren sau în metrou. În Europa, mai puțini oameni se deplasează cu trenul decât în Japonia, iar în SUA abia dacă se deplasează câțiva. Folosirea i-mode acasă, lângă un calculator cu un monitor de 17 țoli, conexiune ADSL de 1 Mbps și toți megaocteții gratuiți, nu se prea justifică. Cu toate acestea, nimeni nu a prezis imensa popularitate a telefoanelor mobile în general, astfel că i-mode mai poate încă să-și găsească o nișă în Vest.

Așa cum am menționat mai sus, telefoanele i-mode folosesc rețeaua cu comutare de circuite existentă pentru voce și o nouă rețea cu comutare de pachete pentru date. Rețeaua pentru date se bazează pe CDMA și transmite pachete de 128 de octeți la 9600 bps. O diagramă a rețelei este dată în fig. 7-50. Telefoanele folosesc **LTP (Lightweight Transport Protocol** – protocol simplificat de transport) pe o legătură prin aer până la o poartă pentru conversie de protocoale. Poarta are o conexiune de bandă largă prin fibră optică la serverul i-mode, care este conectat la toate serviciile. Când utilizatorul selectează un serviciu din meniul oficial, cererea este trimisă serverului i-mode, care ține majoritatea paginilor în memoria ascunsă pentru a-și spori performanța. Cererile pentru situri care nu sunt în meniul oficial ocolesc serverul i-mode și merg direct pe Internet.

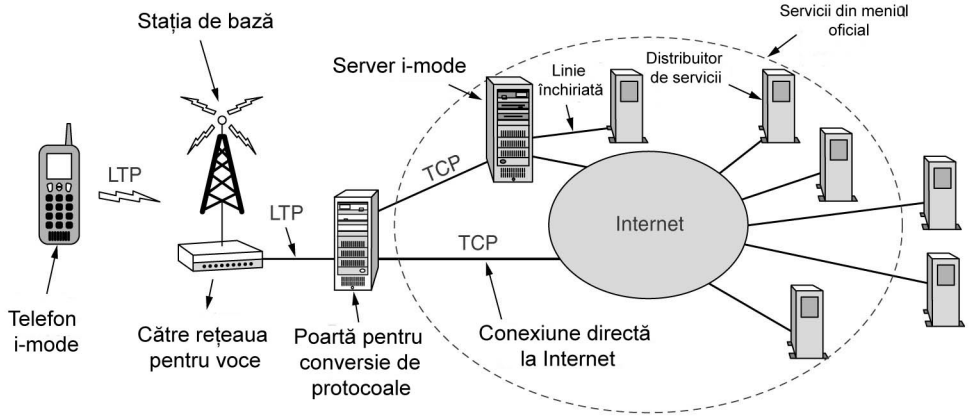


Fig. 7-50. Structura rețelei de date i-mode, arătând protocoalele de transport.

Telefoanele actuale au procesoare care funcționează la aproximativ 100 MHz, câțiva megaocteți de memorie ROM rapidă, poate 1 MB RAM și un ecran mic încorporat. I-mode necesită un ecran de cel puțin 72x94 pixeli, dar unele dispozitive mai mari au chiar 120x160 pixeli. Ecranele au de obicei culori pe 8 biți, ceea ce permite 256 de culori. Aceasta nu este suficient pentru fotografii, dar este adecvat pentru desenarea de linii și imagini animate simple. Cum nu există mouse, navigarea pe ecran se face cu săgețile direcționale.

Modulul de interacțiune cu utilizatorul		
Elemente de intrare	Interpretor cHTML	Java
Coordonator simplu de ferestre		
Comunicație de rețea		
Sistem de operare în timp real		

Fig. 7-51. Structura aplicațiilor i-mode.

Structura aplicațiilor este prezentată în fig. 7-51. Nivelul cel mai de jos conține un sistem simplu de operare în timp real pentru controlul echipamentelor. Apoi vine un modul pentru comunicarea pe rețea, folosind protocolul proprietar al NTT DoCoMo, LTP. Deasupra acestuia vine un simplu coordonator de ferestre care se ocupă de text și de imaginile simple (fișiere GIF). Cu ecranele având doar aproximativ 120x160 de pixeli în cel mai bun caz, nu sunt prea multe de coordonat.

Al patrulea nivel conține interpretorul de pagini Web (de exemplu, programul de navigare). I-mode nu folosește întregul HTML, ci numai un subset al acestuia, numit **cHTML (compact HTML** – HTML compact), bazat în mare pe HTML 1.0. Acest nivel permite de asemenea și aplicații ajutătoare și elemente de intrare, la fel cum fac și programele de navigare pentru PC-uri. O aplicație ajutătoare standard este un interpretor pentru o versiune puțin modificată a JVM.

La nivelul cel mai înalt se află modulul de interacțiune cu utilizatorul, care controlează comunicația cu acesta.

Să ne uităm acum mai în detaliu la cHTML. După cum am menționat, este aproximativ HTML 1.0, cu câteva omisiuni și câteva extensii pentru a fi folosit cu telefoane mobile. A fost trimis la W3C pentru standardizare, dar W3C nu a arătat interes pentru el, așa că probabil va rămâne un produs privat.

Majoritatea etichetelor de bază HTML sunt permise, incluzând aici <html>, <head>, <title>, <body>, <h*n*>, , <ul>, <ol>, <©>, <li>, <br>, <p>, <hr>, <img>, <form> și <input>. Etichetele <b> și <i> nu sunt permise.

Eticheta <a> este permisă pentru legarea la alte pagini, dar cu schema adițională *tel* pentru formarea numerelor de telefon. Dintr-un punct de vedere *tel* este analog cu *mailto*. Când o hiper-legătură folosind schema *mailto* este selectată, programul de navigare deschide un formular pentru a trimite un mesaj electronic către destinația marcată în legătură. Când este selectată o hiper-legătură cu schema *tel*, programul de navigare formează numărul de telefon. Spre exemplu, o agendă de telefon poate conține imagini simple ale unor persoane. Când se selectează una dintre acestea, telefonul îl va suna pe el sau pe ea. RFC 2806 prezintă URL-urile pentru telefoane.

Programul de navigare cHTML este limitat în alte feluri. El nu suportă JavaScript, cadre, foi de stil, culori sau imagini de fundal. De asemenea, nu suportă imagini JPEG, deoarece acestea se decomprimă în prea mult timp. Sunt permise applet-urile Java, dar sunt (în prezent) limitate la 10 KB din cauza vitezei mici de transmisie a legăturii prin aer.

Deși NTT DoCoMo a eliminat câteva etichete HTML, a și adăugat unele noi. Eticheta <blink> face ca textul să se afișeze și apoi să dispară. Deși pare ciudat să interzici eticheta <b> (motivând că siturile Web nu ar trebui să se ocupe de prezentarea conținutului) și apoi să adaugi <blink> care nu se referă decât la prezentarea conținutului, asta au făcut. O altă etichetă nouă este <marquee>, care își deplasează conținutul pe ecran precum un monitor de bursă.

Un element nou este atributul *align* al etichetei <br>. Este necesar, deoarece pentru un ecran ce are de obicei 6 rânduri de câte 16 caractere, există un mare pericol ca rândurile să fie rupte în două, ca în fig. 7-52(a). *Align* ajută la reducerea acestei probleme și conduce la ceva ce seamănă mai degrabă cu fig. 7-52(b). Este interesant să notăm că limba japoneză nu suferă când cuvintele sunt rupte între linii. Pentru textul kanji, ecranul este împărțit într-un caroiaj dreptunghiular de celule de dimensiune 9x10 pixeli sau 12x12 pixeli, în funcție de caracterele suportate. Fiecare celulă conține exact un caracter kanji, care este echivalentul unui cuvânt în engleză. Despărțirea mai multor cuvinte pe mai multe linii este întotdeauna admisă în japoneză.

(a) (b)

Fig. 7-52. Lewis Caroll începe într-un ecran de 16x6.

Deși limba japoneză are zeci de mii de kanji, NTT DoCoMo a inventat 166 noi, numite **emoji**, cu un factor de amuzament ridicat – de fapt, niște pictograme precum zâmbitorii din fig. 7-6. Acestea includ simboluri pentru semnele astrologice, bere, hamburger, parc de amuzament, zi de naștere, telefon mobil, câine, pisica, Crăciun, inimă rănită, sărut, stare de spirit, somnolență, și desigur, unul semnificând ceva simpatic.

Un alt nou element este posibilitatea de a permite utilizatorilor să selecteze hiper-legături folosind tastatura, proprietate cu siguranță importantă pentru un calculator fără mouse. Un exemplu despre cum este folosit acest atribut este prezentat în fișierul cHTML din fig. 7-53.

```
<html>
<body>
<h1> Selectează o opțiune </h1>
<a href="messages.chtml" accesskey="1"> Verifică poșta vocală </a> <br>
<a href="mail.chtml" accesskey="2"> Verifică poșta electronică </a> <br>
<a href="games.chtml" accesskey="3"> Joacă un joc </a>
</body>
</html>
```

Fig. 7-53. Un exemplu de fișier cHTML.

Deși partea client este oarecum limitată, serverul i-mode este un calculator de-a dreptul răsunător, cu toate soneriile și fluierele obișnuite. El suportă CGI, Perl, PHP, JSP, ASP și tot ceea ce serverele Web suportă de obicei.

Caracteristica	**WAP**	**I-mode**
Ce este	Stivă de protocoale	Serviciu
Dispozitiv	Telefon, PDA, calculator portabil	Telefon
Tipul de acces	Prin telefon	Tot timpul activ
Rețeaua de suport	Cu comutare de circuite	Doua: circuite + pachete
Rata de transfer	9600 bps	9600 bps
Ecranul	Monocrom	Color
Limbajul de marcare	WML (aplicație XML)	cHTML
Limbajul script	WMLScript	Nici unul
Taxarea utilizatorilor	Pe minut	Pe pachet
Plata pentru cumpărături	Cu cartea de credit	Pe factura de telefon
Pictograme	Nu	Da
Standardizare	Standard deschis al forumului WAP	Proprietar NTT DoCoMo
Locul de utilizare	Europa, Japonia	Japonia
Utilizatorul tipic	Omul de afaceri	Adolescentă

Fig. 7-54. O comparație între prima generație de WAP și i-mode.

O comparație rapidă între WAP și i-mode așa cum au fost ele implementate în sistemele de prima generație este dată în fig. 7-54. În timp ce câteva diferențe pot părea mici, adesea ele sunt importante. Spre exemplu, cei de 15 ani nu au cărți de credit, astfel că posibilitatea de a cumpăra lucruri prin comerțul electronic și de a fi taxați abia când primesc nota de plată la telefon reprezintă un stimulent pentru interesul lor asupra sistemului.

Pentru alte informații despre i-mode citiți (Frengle, 2002; și Vacca, 2002).

Web-ul fără fir de generația a doua

WAP 1.0, bazat pe standarde recunoscute internațional, trebuia să fie o unealtă importantă pentru oamenii cu afaceri în derulare. A eșuat. I-mode a fost o jucărie electronică pentru adolescenții japonezi folosind numai elemente proprietare. A fost un mare succes. Ce s-a întâmplat după asta? Fiecare parte a învățat câte ceva din Web-ul fără fir de primă generație. Consorțiul WAP a învățat că important este conținutul. Să nu ai un număr mare de situri Web care îți înțeleg limbajul de marcare este fatal. NTT DoCoMo a învățat că un sistem închis, proprietar, strâns legat de dispozitive mici și de cultura japoneză nu este un bun produs pentru export. Concluzia pe care ambele părți au tras-o este că pentru a convinge un număr mare de situri Web să-și pună conținutul în formatul tău, trebuie să ai un limbaj de marcare deschis, stabil și care este universal acceptat. Războaiele asupra formatelor nu sunt bune pentru progres.

Ambele servicii sunt aproape de a intra în cea de-a doua generație a tehnologiei Web fără fir. WAP 2.0 a venit primul, așa că îl vom folosi pe acesta ca exemplu. WAP 1.0 avea unele lucruri bune și acestea au fost continuate. Unul dintre acestea este că WAP poate folosi o mulțime de rețele diferite. Prima generație folosea rețele cu comutare de circuite, dar rețelele cu comutare de pachete au fost întotdeauna o alternativă și încă mai sunt. Sistemele de a doua generație vor folosi probabil comutarea de pachete, spre exemplu, GPRS-ul. Un altul este că WAP a fost inițial proiectat pentru a suporta o mare varietate de dispozitive, de la telefoane mobile până la calculatoare portabile puternice, și încă mai este.

WAP 2.0 are și câteva caracteristici noi. Cele mai importante sunt:

1. Modelul de livrare a paginilor (push), alături de cel de cerere (pull).
2. Suport pentru integrarea telefoniei în aplicații.
3. Mesageria multimedia.
4. Includerea a 264 de pictograme.
5. Interfața cu un dispozitiv de memorare.
6. Suport pentru plug-in-uri în browser.

Modelul de cerere este bine cunoscut: clientul cere o pagină și o primește. Modelul de livrare (push) suportă livrarea datelor fără a fi cerute, precum ținerea la curent cu cotațiile la bursă sau alertele de trafic.

Vocea și datele încep să se contopească, iar WAP 2.0 le suportă într-o mulțime de moduri. Am văzut un astfel exemplu mai devreme, la capabilitatea i-mode-ului de a lega o iconiță sau un text de pe ecran cu un număr de telefon ce trebuie format. Odată cu poșta electronică și telefonia este suportată și mesageria multimedia.

Marea popularitate a emoji-ului i-mode a stimulat consorțiul WAP să inventeze 264 emoji proprii. Categoriile includ animale, utilaje casnice, îmbrăcăminte, emoții, mâncăruri, corpul uman, genuri, hărți, muzică, plante, sporturi, timp, unelte, vehicule, arme și meteo. Destul de interesant este faptul că standardul pur și simplu numește fiecare pictogramă; nu dă harta de pixeli reală, probabil

de teamă că reprezentarea într-o cultură a „somnolenței" sau a „îmbrățișării" să nu insulte altă cultură. I-mode nu a avut această problemă fiind îndreptat către o singură țară.

A oferi o interfață de stocare nu înseamnă că fiecare telefon cu WAP 2.0 va veni cu un mare disc fix. Memoria ROM rapidă este, de asemenea, un dispozitiv de stocare. O cameră fără fir cu facilități WAP ar putea folosi memoria ROM rapidă pentru stocarea temporară a imaginii înainte de a transmite cele mai bune cadre pe Internet.

În fine, plug-in-urile pot extinde capabilitățile programului de navigare. Este oferit, de asemenea, un limbaj scriptic.

Mai multe diferențe tehnice sunt de asemenea prezente în WAP 2.0. Două dintre cele mai importante se referă la stiva de protocoale și la limbajul de marcare. WAP 2.0 continuă să suporte vechea stivă de protocoale din fig. 7-48, dar de asemenea suportă și stiva standard a Internet-ului cu TCP și ©/1.0. Cu toate acestea, patru schimbări minore (dar compatibile) au fost aduse TCP-ului (pentru a simplifica codul): (1) Folosirea unei ferestre fixe de 64 KB, (2) lipsa unui start lent, (3) MTU-ul maxim de 1500 de octeți și (4) un algoritm de retransmisie ușor modificat. TLS este protocolul pentru securitatea la nivel transport standardizat de IETF; îl vom examina în Cap. 8. Mai multe dispozitive inițiale vor conține probabil ambele stive, cum se arată în fig. 7-55.

Fig. 7-55. WAP 2.0 suportă două stive de protocoale.

Cealaltă diferență tehnică față de WAP 1.0 este limbajul de marcare. WAP 2.0 suportă XHTML Basic, care este potrivit pentru dispozitivele mici fără fir. Întrucât NTT DoCoMo a fost de asemenea de acord să suporte acest subset, proiectanții de situri Web pot folosi acest format știind că paginile lor vor funcționa atât pe Internet-ul fix cât și pe toate dispozitivele fără fir. Aceste decizii vor încheia războaiele legate de formatul limbajului de marcare care au împiedicat dezvoltarea industriei Web fără fir.

Modulul	Obligatoriu?	Funcția	Exemple de etichete
Structură	Da	Structura documentelor	body, head, html, title
Text	Da	Informații	br, code, dfn, em, h*n*, kbd, p, strong
Hiper-text	Da	Hiper-legături	a
Liste	Da	Liste de articole	dl, dt, dd, ol, ul, li
Formulare	Nu	Formulare de completat	form, input, label, option, textarea
Tabele	Nu	Tabele dreptunghiulare	caption, table, td, th, tr
Imagini	Nu	Imagini	img
Obiecte	Nu	Applet-uri, hărți, etc.	object, param
Meta-informații	Nu	Informații suplimentare	meta
Legături	Nu	Similar cu <a>	link
Bază	Nu	URL-ul de start	base

Fig. 7-56. Modulele și etichetele din XHTML Basic.

Câteva cuvinte despre XHTML Basic sunt poate necesare. Acesta este gândit pentru telefoane mobile, televiziune, PDA-uri, dispozitive pentru vânzarea automată, pagere, mașini, jocuri mecanice și chiar ceasuri. Din acest motiv nu suportă foi de stil, scripturi sau cadre, însă cunoaște majoritatea etichetelor standard. Acestea sunt grupate în 11 module. Unele sunt obligatorii; altele sunt opționale. Toate sunt definite în XML. Modulele și câteva exemple de etichete sunt listate în fig. 7-56. Nu baleiat toate exemplele de etichete, însă mai multe informații se pot găsi la *www.w3.org*.

În ciuda înțelegerii asupra folosirii XHTML Basic, o amenințare pândește WAP și i-mode: 802.11. A doua generație a Web-ului fără fir ar trebui să funcționeze la 384 Kbps, mult mai mult decât cei 9600 bps ai primei generații, dar și mult mai puțin decât cei 11 Mbps sau 54 Mbps oferiți de 802.11. Firește, 802.11 nu este omniprezent, dar pe măsură ce mai multe restaurante, hoteluri, magazine, companii, aeroporturi, stații de autobuz, muzee, universități, spitale și alte organizații decid să instaleze stații de bază pentru angajații și clienții lor, se va ajunge probabil la o suficientă acoperire în zonele urbane astfel încât oamenii să dorească să meargă pentru o cafea sau pentru a trimite un mesaj electronic la o braserie aflată la câteva blocuri distanță, dar cu 802.11 instalat. Firmele pot adăuga automat embleme 802.11 alături de emblemele care arată ce cărți de credit acceptă și asta din același motiv: pentru a atrage clienți. Hărțile orașelor (firește, descărcabile) pot marca zonele acoperite cu verde și pe cele neconectate cu roșu, astfel ca oamenii să se poată deplasa de la o stație de bază la alta, așa cum nomazii se mutau de la o oază la alta în deșert.

Deși braseriile pot instala repede stații de bază 802.11, fermierilor probabil că nu le va fi la fel de ușor, deci acoperirea va fi zonală și limitată la zonele centrale ale orașelor, din cauza răspândirii limitate a semnalului 802.11 (câteva sute de metri în cel mai bun caz). Aceasta poate duce la dispozitive fără fir cu două moduri, care folosesc 802.11 dacă pot prinde un semnal și recurg la WAP dacă nu.

7.4 MULTIMEDIA

Deși Web-ul fără fir este o tehnologie nouă si incitantă, ea nu este singura. Pentru mulți, multimedia reprezintă sarea și piperul rețelelor de calculatoare. Mințile ascuțite văd imense provocări tehnice în furnizarea de video (interactiv) la cerere în fiecare casă. Gulerele albe văd un profit imens în acestea. Întrucât multimedia necesită o lățime de bandă mare, este destul de greu să fie făcută să funcționeze prin conexiuni fixe. Chiar și calitatea video VHS prin legătura fără fir este la distanță de câțiva ani, așa că discuția noastră se va axa asupra sistemelor conectate.

Literal, multimedia înseamnă două sau mai multe media. Dacă editorul acestei cărți voia să se alăture interesului la modă despre multimedia, el putea anunța că lucrarea folosește tehnologia multimedia. În fond, aceasta conține două media: textul și grafica (desenele). Cu toate acestea, atunci când majoritatea oamenilor se referă la multimedia, de fapt ei se referă la combinarea între două sau mai multe **media continue** (continuous media), adică media care trebuie să se desfășoare într-un interval bine definit, de obicei folosind interacțiunea cu utilizatorul. În practică, cele două media sunt audio și video, adică sunete plus filme.

Oricum, mulți vorbesc adesea despre sunetele audio pure, precum telefonia prin Internet sau radio-ul prin Internet, ca și cum ar fi tot multimedia, ceea ce cu siguranță nu sunt. De fapt, un termen mai bun ar fi **fluxuri media** (streaming media), dar vom urma mulțimea și vom considera sunetele audio în timp real ca fiind tot multimedia. În secțiunile următoare vom examina modul în care calcu-

latoarele procesează datele audio și video, cum le comprimă, și câteva aplicații pentru rețele ale acestor tehnologii. Pentru o tratare cuprinzătoare (trei volume) a datelor multimedia în rețele, citiți (Steinmetz si Nahrstedt, 2002; Steinmetz si Nahrstedt, 2003a; și Steinmetz și Nahrstedt, 2003b).

7.4.1 Introducere in sunetele digitale

O undă (sunet) audio este o undă cu o dimensiune acustică (presiune). Atunci când o undă acustică intră în ureche, pavilionul vibrează, făcând ca oasele urechii interne să vibreze o dată cu el, transmițând vibrații nervoase creierului. Aceste vibrații sunt percepute drept sunete de către ascultător. Într-un mod similar, atunci când o undă acustică lovește un microfon, acesta generează un semnal electric, reprezentând amplitudinea sunetului ca funcție de timp. Reprezentarea, procesarea, memorarea, și transmisia acestor semnale audio constituie părțile majore ale studiului sistemelor multimedia.

Intervalul de frecvență pentru urechea umană este cuprins între 20 Hz și 20.000 Hz, deși unele animale, în special câinii, pot percepe frecvențe mai înalte. Urechea percepe sunetele în mod logaritmic, așa încât raportul a două semnale cu puterile A și B este exprimat convențional în **dB (decibeli)** în concordanță cu formula:

$$\text{dB} = 10 \log_{10} (A/B)$$

Dacă definim limita inferioară de audibilitate (o presiune de circa 0,0003 dyne/cm^2) pentru o undă sinusoidală de 1 KHz ca 0 dB, o conversație obișnuită este în jur de 50 dB și pragul de durere este în jur de 120 dB, un interval dinamic cu un factor de 1 milion. Pentru a evita orice confuzie, A și B de mai sus sunt numite *amplitudini*. Dacă trebuie să folosim nivelul puterii, care este proporțional cu pătratul amplitudinii, coeficientul logaritmului va fi 10, nu 20.

Urechea este surprinzător de sensibilă la variații ale sunetului care durează doar câteva milisecunde. Ochiul, în schimb, nu poate observa schimbările de nivel ridicat care durează doar câteva milisecunde. Rezultatul acestei observații constă în faptul că o variație de numai câteva milisecunde din timpul unei transmisii multimedia afectează calitatea sunetului perceput mai mult decât afectează calitatea imaginii percepute.

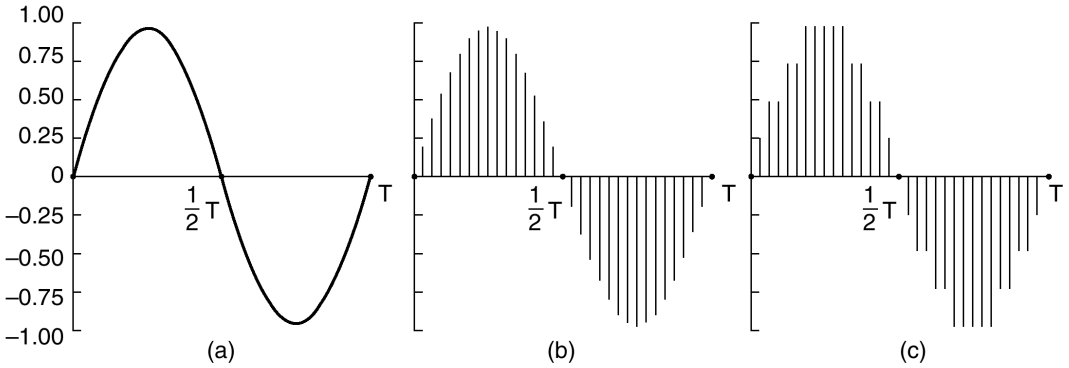


Fig. 7-57. (a) O undă sinusoidală. (b) Eșantionarea undei sinusoidale. (c) Cuantificarea eșantioanelor pe 4 biți.

Undele audio pot fi convertite în formă numerică de un **ADC (Analog Digital Convertor –** convertor analog numeric). Un ADC primește o tensiune electrică la intrare și generează un număr binar la ieșire. În fig. 7-57(a) se prezintă un exemplu al unei unde sinusoidale. Pentru reprezentarea acestui semnal în formă digitizată, putem să-l eșantionăm la fiecare ΔT secunde, așa cum puteți observa prin înălțimea barelor în fig. 7-57(b). Dacă o undă sonică nu este o undă pur sinusoidală, ci o superpoziție liniară de unde sinusoidale, unde cea mai mare componentă a frecvenței prezente este *f*, atunci teorema lui Nyquist (vezi cap. 2) spune că este suficient să se eșantioneze la frecvența *2f*. Eșantionând mai des, nu se obține nimic în plus, deoarece frecvențele mai înalte pe care discretizarea le poate detecta nu sunt prezente.

Eșantioanele digitale nu sunt niciodată exacte. Eșantioanele pe 3-biți din fig. 7-57© permit doar opt valori, de la -1,00 la +1,00 în pași de 0,25. Un eșantion pe 8-biți permite 256 valori distincte. Un eșantion pe 16-biți admite 65.536 valori distincte. Eroarea introdusă de numărul finit de biți de eșantionare se numește **zgomot de cuantizare** (quantization noise). Dacă este foarte mare, urechea îl detectează.

Două exemple bine cunoscute de sunete eșantionate sunt telefonul și compact discurile audio. Modularea impulsului în cod, folosită în sistemul telefonic, utilizează eșantioane pe 8 biți de 8000 de ori pe secundă. În America de Nord și Japonia, 7 biți sunt pentru date, iar unul pentru control; în Europa toți cei 8 biți sunt pentru date. Acest sistem furnizează o viteză a datelor de 56.000 bps sau 64.000 bps. Cu doar 8000 de eșantioane/sec, frecvențele peste 4 KHz sunt pierdute.

CD-urile audio sunt digitale, cu o rată de eșantionare de 44.100 eșantioane/sec, suficientă pentru a putea capta frecvențele până la 22.050 Hz, care sunt bune pentru oameni, dar rele pentru iubitorii canini de muzică. Eșantioanele au fiecare 16 biți și sunt liniare în domeniul amplitudinii. Observați că eșantioanele pe 16-biți permit doar 65.536 valori distincte, chiar dacă limita dinamică a urechii este de 1 milion atunci când se măsoară în unități de cel mai mic sunet audibil. Astfel, folosind doar 16 biți pe eșantion, se introduce un zgomot de cuantizare (deși întreaga rată dinamică nu este acoperită – CD-urile se presupune că nu rănesc). Cu 44.100 eșantioane/sec pe 16 biți fiecare, un CD audio are nevoie de o lărgime de bandă de 705,6 Kbps pentru mono și de 1,411 Mbps pentru stereo. Deși acesta este mai scăzut decât necesită video-ul (vezi mai jos), poate acapara aproape un canal întreg T1 pentru transmiterea necomprimată a sunetului stereo de calitate pentru CD în timp real.

Sunetele digitizate pot fi procesate ușor de calculatoare, prin programe. Există zeci de programe pentru calculatoarele personale care permit utilizatorilor să înregistreze, să afișeze, să editeze, să amestece și să memoreze undele sonore de la surse multiple. Practic, toate înregistrările, editările de sunete profesioniste sunt în prezent digitale.

Muzica este desigur un caz special de audio, dar unul important. Alt caz important este discursul. Discursul uman tinde să fie între limitele de 600 Hz și 6000 Hz. Discursul este format din vocale și consoane, care au proprietăți diferite. Vocalele sunt produse atunci când coardele vocale sunt neobstrucționate, producând rezonanțe ale căror frecvențe fundamentale depind de dimensiunea și forma sistemului vocal și de poziția limbii vorbitorului și a gurii. Aceste sunete sunt aproape periodice pentru intervale în jur de 30 ms. Consoanele sunt produse atunci când coarda vocală este parțial blocată. Aceste sunete sunt mai puțin regulate decât vocalele.

Câteva sisteme de generare și transmisie de voce pot folosi modelele de sisteme vocale pentru a reduce vocea la câțiva parametri (de exemplu, mărimile și formele diferitelor cavități), mai degrabă decât de a eșantiona forma de undă pentru voce. Oricum, modul de funcționare al acestor codoare de voce depășește domeniul acestei cărți.

7.4.2 Compresia audio

Pentru a obține calitatea unui CD audio este necesară o lățime a benzii de transmisie de 1.411 Mbps, după cum tocmai am văzut. Evident, pentru a face practică transmisia pe Internet este necesară o compresie substanțială. Mulți algoritmi de compresie audio au fost dezvoltați din acest motiv. Probabil cel mai popular dintre ele este MPEG audio, care are trei nivele (variante), dintre care **MP3 (MPEG audio layer 3** – MPEG audio de nivelul 3) este cel mai puternic și mai cunoscut. Pe Internet sunt disponibile mari cantități de muzică în format MP3, nu toate legale, fapt care a generat numeroase procese venite din partea artiștilor și a proprietarilor de drepturi de autor. MP3 aparține părții audio a standardului MPEG pentru compresia video. Vom discuta despre compresia video mai târziu în acest capitol; să vedem acum compresia audio.

Compresia audio poate fi făcută în două moduri. Prin **codificarea formei de undă** (waveform coding)**,** semnalul este transformat (matematic) Fourier în componentele sale în frecvență. Fig. 2-1(a) arată un exemplu de funcție de timp și amplitudinile sale Fourier. Apoi, amplitudinea fiecărei componente este codificată minimal. Scopul este de a reproduce exact forma de undă la celălalt capăt folosind cât mai puțini biți cu putință.

Cealaltă metodă, **codificarea perceptivă** (eng. Perceptual coding), exploatează unele caracteristici ale sistemului auditiv uman pentru a codifica semnalul astfel încât să sune la fel pentru ascultătorul uman, chiar dacă arată destul de diferit pe osciloscop. Codificarea perceptivă se bazează pe știința **psihoacusticii** – modul în care oamenii percep sunetul. MP3 se bazează pe codificarea perceptivă.

Proprietatea de bază a codificării perceptive este că unele sunete pot **masca** alte sunete. Imaginați-vă că transmiteți în direct un concert de flaut într-o zi călduroasă de vară. Apoi, dintr-o dată, un grup de muncitori din apropiere își pornesc ciocanele pneumatice și încep să distrugă strada. Nimeni nu mai poate auzi flautul. Sunetele sale au fost mascate de ciocanele pneumatice. Din punct de vedere al transmisiei, acum este suficient să codificăm doar banda de frecvențe folosită de ciocanele pneumatice, deoarece ascultătorii oricum nu pot auzi flautele. Acest procedeu se numește **mascarea frecvenței** (frequency masking) – proprietatea unui sunet de volum înalt dintr-o bandă de frecvență de a ascunde un sunet mai slab dintr-o altă bandă de frecvență și care s-ar fi auzit în absența sunetului de volum înalt. De fapt, chiar și după ce ciocanele pneumatice încetează, flautul nu se va putea auzi pentru o scurtă perioadă de timp, deoarece urechile opresc amplificarea sunetelor când acestea încep și au nevoie de o perioadă de timp finită pentru a o reporni. Acest efect se numește **mascare temporală** (temporal masking).

Pentru a cuantifica aceste efecte, imaginați-vă experimentul 1. O persoană dintr-o cameră în care este liniște își pune căștile conectate la placa de sunet a unui calculator. Calculatorul generează o undă sinusoidală nedistorsionată la 100 Hz la o putere mică inițial dar crescătoare în timp. Persoana este instruită să apese pe o tastă când aude sunetul. Calculatorul înregistrează nivelul curent al puterii și repetă experimentul la 200 Hz, 300 Hz și toate celelalte frecvențe până la limita auzului uman. Când se face o medie pentru mai mulți oameni, un grafic al înregistrărilor puterii necesare unui sunet pentru a fi auzit arată ca cel din fig. 7-58(a). O consecință directă a acestei curbe este că nu trebuie să codificăm frecvențele a căror putere se află sub pragul auzului. De exemplu, dacă puterea la 100 Hz ar fi fost 20 dB în fig. 7-58(a), ar fi putut fi omisă din sunetul final fără o pierdere sesizabilă de calitate, deoarece 20 dB la 100 Hz sunt sub nivelul de audibilitate.

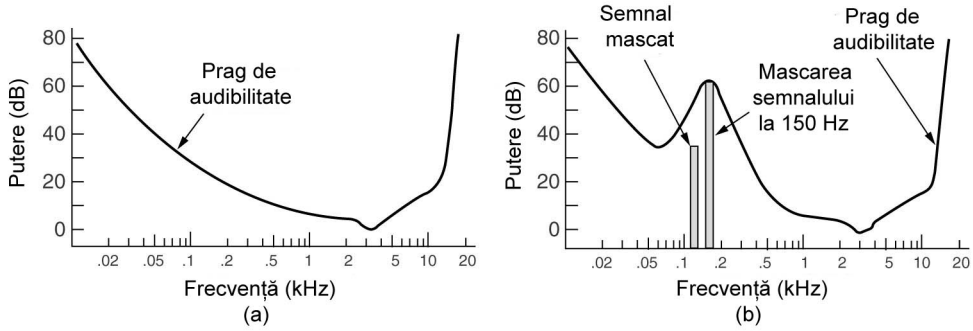


Fig. 7-58. (a) Pragul de audibilitate ca funcție de frecvență. (b) Efectul de mascare.

Acum imaginați-vă experimentul 2. Calculatorul rulează din nou experimentul 1, dar de această dată cu o undă sinusoidală de amplitudine constantă și frecvența 150 Hz, suprapusă peste frecvența de test. Descoperim că pragul de audibilitate pentru frecvențe de aproximativ 150 Hz este ridicat, așa cum arată și fig. 7-58(b).

Consecința acestei noi observații este că, pe măsură ce descoperim care semnale sunt mascate de alte semnale din benzi de frecvență apropiate, putem omite mai multe frecvențe din semnalul codificat, economisind biți. În fig. 7-58, semnalul de 125 Hz poate fi în întregime omis din semnalul de ieșire și nimeni nu va putea sesiza diferența. Chiar și după ce un semnal puternic dintr-o bandă de frecvență se oprește, cunoașterea proprietăților sale de mascare temporală ne permit să continuăm să omitem frecvențele mascate pentru un anumit interval de timp până când urechea își revine. Elementul de bază al MP3 este transformarea Fourier a sunetului pentru a obține puterea pentru fiecare frecvență și apoi transmiterea numai a frecvențelor nemascate, codificându-le în cât mai puțini biți cu putință.

Cunoscând aceste informații, putem acum să vedem cum se face codificarea. Compresia audio se face eșantionând forma de undă la 32 KHz, 44.1 KHz sau 48 KHz. Eșantionarea se poate face pe unul sau două canale, în una din cele patru configurații:

1. Monofonic (un singur flux de intrare).
2. Dual monofonic (spre exemplu, o coloană sonoră in engleză și una în japoneză).
3. Stereo disjunct (fiecare canal comprimat separat).
4. Stereo unit (se exploatează la maxim redundanța inter-canale).

Mai întâi se alege rata biților de la ieșire. MP3 poate comprima un CD rock'n'roll stereo până la minim 96 Kbps cu pierderi de calitate greu perceptibile, chiar și pentru fanii rock'n'roll care nu au probleme cu auzul. Pentru un concert de pian sunt necesari cel puțin 128 Kbps. Acestea diferă deoarece raportul semnal-zgomot pentru rock'n'roll este mult mai mare față de cel al unui concert de pian (cel puțin din punct de vedere ingineresc). Este de asemenea posibil să alegem rate de ieșire mai mici acceptând o pierdere în calitate.

Apoi eșantioanele sunt procesate în grupuri de 1152 (lungi de aproximativ 26 ms). Fiecare grup este întâi trecut prin 32 de filtre digitale pentru a obține 32 de benzi de frecvență. În același timp, intrarea

este dată unui model psiho-acustic care determină frecvențele mascate. Apoi, fiecare din cele 32 de benzi de frecvență este transformată în continuare pentru a obține o rezoluție spectrală mai fină.

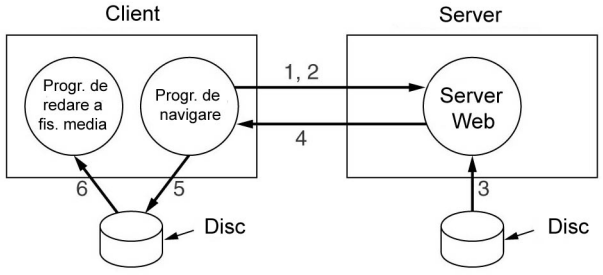
În faza următoare, bugetul de biți disponibil este împărțit între benzi, cât mai mulți biți alocați pentru benzile cu puterea spectrală nemascată mai mare, cât mai puțini biți alocați pentru benzile nemascate cu puterea spectrală mai mică și nici un bit alocat pentru benzile mascate. În final, biții sunt codificați folosind codificarea Huffman, care asignează coduri scurte numerelor care apar frecvent și coduri lungi celor care apar rar.

În realitate mai sunt multe de spus. Se folosesc de asemenea multe tehnici pentru reducerea zgomotelor, antialiasing și exploatarea redundanței inter-canale, dacă este posibil, dar acestea sunt în afara domeniului acestei cărți. O introducere matematică mai formală în această operație este dată în (Pan, 1995).

7.4.3 Fluxuri audio

Să trecem acum de la tehnologia audio digitală la trei dintre aplicațiile sale pentru rețele. Prima este fluxul audio, adică ascultarea sunetelor pe Internet. Aceasta se numește de asemenea și muzică la cerere. Celelalte două sunt radio-ul prin Internet și respectiv vocea peste IP.

Internet-ul este plin de situri Web cu muzică, multe dintre ele listând titluri de cântece pe care utilizatorii le pot selecta cu ajutorul mouse-ului pentru le asculta. Unele dintre acestea sunt situri gratuite (spre exemplu formațiile noi care încearcă să își facă publicitate); altele necesită o plată pentru muzică, deși ele oferă de asemenea mostre gratuite (spre exemplu primele 15 secunde dintr-un cântec). Metoda cea mai rapidă de a asculta muzica este ilustrată în fig. 7-59.



1. Stabilirea conexiunii TCP
2. Trimiterea cererii HTTP GET
3. Serverul preia fișierul de pe disc
4. Fișierul este trimis înapoi programului de navigare
5. Programul de navigare scrie fișierul pe disc
6. Programul pentru redarea fișierelor media preia fișierul bloc după bloc și îl redă

Fig. 7-59. O metodă directă pentru a implementa muzica selecționabilă de pe o pagină Web.

Procesul începe când utilizatorul selectează cu mouse-ul un cântec. Apoi intră în acțiune programul de navigare. Primul său pas este stabilirea unei conexiuni TCP cu serverul Web pe care există o hiper-legătură la cântec. Pasul 2 este trimiterea unei cereri *GET* în HTTP pentru a cere cântecul. Ulterior (pașii 3 si 4), serverul citește cântecul (care este doar un fișier în formatul MP3 sau vreun alt format) de pe disc și îl trimite înapoi programului de navigare. Dacă fișierul este mai mare decât memoria serverului, acesta poate aduce și trimite melodia în blocuri.

Folosind tipul MIME, de exemplu, *audio/mp3*, (sau extensia fișierului), programul de navigare caută să vadă cum ar trebui livrat fișierul. De obicei există o aplicație ajutătoare asociată cu acest tip de fișier, precum RealOne Player, Windows Media Player, sau Winamp. Întrucât în mod obișnuit programul de navigare comunică cu o aplicație ajutătoare printr-un fișier auxiliar, acesta va salva mai întâi întregul fișier de muzică pe disc ca un fișier auxiliar (pasul 5). Apoi, va porni programul de re-

dare a fișierului căruia îi va trimite numele fișierului auxiliar de pe disc. La pasul 6, programul pentru redarea fișierelor media va începe să încarce și să reproducă muzica, bloc după bloc.

În principiu, această metodă este în întregime corectă și va produce muzică. Singura problemă este că trebuie trimis prin rețea tot cântecul, înainte ca muzica să înceapă. În cazul în care cântecul are 4 MB (o dimensiune tipică pentru un cântec MP3) și modemul este de 56 Kbps, utilizatorul va avea parte de aproape 10 minute de liniște până când cântecul este descărcat. Nu toți îndrăgostiții de muzică agreează această idee. Mai ales având în vedere că următorul cântec va începe tot după 10 minute de descărcare, iar cel de după acesta la fel.

Pentru a rezolva această problemă fără a schimba felul în care funcționează programul de navigare, siturile cu muzică au venit cu următoarea schemă. Fișierul legat la titlul cântecului nu este fișierul real cu muzică. În schimb, este ceea ce se numește un **metafișier**, adică un fișier foarte scurt cu numele melodiei. Un metafișier tipic poate avea doar o singură linie de text ASCII și arată astfel:

```
rtsp://joes-audio-server/song-0025.mp3
```

Când programul de navigare primește fișierul de o linie, îl scrie pe disc într-un fișier auxiliar, pornește programul pentru redarea fișierelor media ca pe o aplicație ajutătoare și îi trimite acestuia numele fișierului auxiliar, ca de obicei. Programul pentru redarea fișierelor media citește fișierul și vede că acesta conține un URL. Apoi contactează *joes-audio-server* și cere cântecul. Observați că programul de navigare nu mai face parte din circuit.

În cele mai multe cazuri, serverul numit în metafișier nu este același cu serverul Web. De fapt, de obicei nu este nici măcar un server HTTP, ci un server specializat pe media. În acest exemplu, serverul media folosește **RTSP** (**Real Time Streaming Protocol** - protocolul pentru fluxuri în timp real), indicat de numele *rtsp* al schemei. Acesta este descris în RFC 2326.

Programul de redare a fișierelor media are patru lucruri importante de făcut:

1. Controlează interfața cu utilizatorul.
2. Tratează erorile de transmisie.
3. Decomprimă melodia.
4. Elimină fluctuațiile.

Majoritatea programelor de redare a fișierelor media din zilele noastre au o interfață captivantă cu utilizatorul, uneori simulând o unitate stereo, cu butoane, mânere, glisoare și afișaje vizuale. Adesea există panouri frontale interschimbabile, numite **învelitori**, pe care utilizatorul le poate suprapune peste panoul programului de redare. Programul de redare trebuie să controleze toate acestea și să interacționeze cu utilizatorul.

A doua funcție a sa este tratarea erorilor. Transmisia de muzică în timp real folosește rareori TCP deoarece o eroare și o retransmisie pot introduce o pauză inacceptabil de lungă în melodie. În locul acestuia, transmisia reală se face de obicei cu un protocol asemănător cu RTP, pe care l-am studiat în Cap. 6. Ca majoritatea protocoalelor de timp real, RTP utilizează UDP și deci se pot pierde pachete. Programul de redare este cel care trebuie să trateze acest lucru.

În unele cazuri, muzica este întrețesută pentru a face tratarea erorilor mai ușoară. Spre exemplu, un pachet poate conține 220 de eșantioane stereo, fiecare conținând o pereche de numere de 16 biți, de obicei bune pentru 5 ms de muzică. Dar protocolul poate trimite toate eșantioanele impare pentru un interval de 10 ms într-un pachet și toate eșantioanele pare în următorul. Atunci un pachet pierdut nu reprezintă o pauză de 5 ms în muzică, ci pierderea tuturor celorlalte eșantioane pentru 10 ms. Această pierdere poate fi tratată ușor de programul de redare a fișierelor media interpolând eșantioanele anterioare și următoare pentru a estima valoarea absentă.

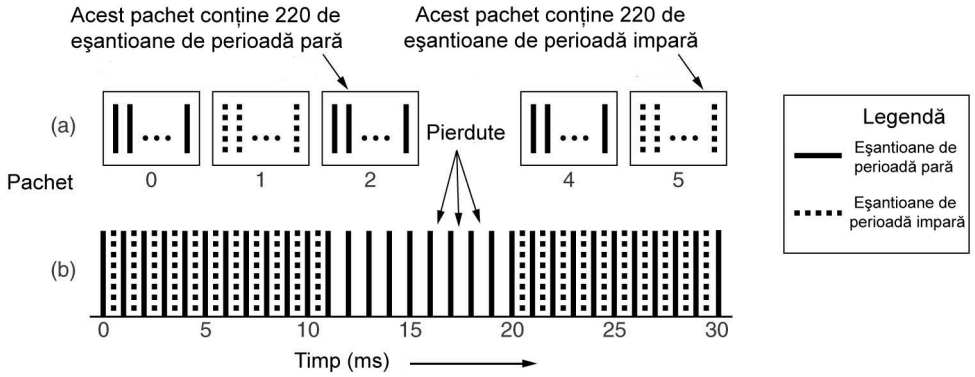


Fig. 7-60. Când pachetele conțin eșantioane alternante, pierderea unui pachet reduce temporar rezoluția în loc să creeze o perioadă de pauză.

Folosirea întrețeserii pentru a recupera din erori este ilustrată în fig. 7-60. Aici fiecare pachet reține eșantioanele alternante în timp pentru un interval de 10 ms. În consecință, pierderea pachetului 3, după cum este arătat, nu introduce o pauză în muzică, ci doar scade rezoluția pentru un anume interval. Valorile absente pot fi interpolate pentru a oferi sunet continuu. Această schemă particulară nu funcționează decât cu eșantioane necomprimate, dar arată cum o codificare inteligentă poate converti un pachet pierdut într-unul de calitate mai joasă, mai degrabă decât într-o pauză de timp. În orice caz, RFC 3119 dă o schemă care funcționează cu date audio comprimate.

A treia funcție a programului de redare a fișierelor media este decomprimarea melodiei. Deși această operație necesită multe resurse, ea este relativ rapidă.

A patra funcție este eliminarea fluctuațiilor, blestemul tuturor sistemelor în timp real. Toate sistemele de fluxuri audio încep prin a depune într-un tampon 10-15 sec. de muzică înainte de a începe să cânte, ca în fig. 7-61. Ideal, serverul va continua să umple tamponul cu aceeași viteză cu care este golit de programul de redare a fișierelor media, însă în realitate nu se întâmplă așa, astfel că se poate recurge la umplerea tamponului în interiorul buclei.

Două soluții pot fi adoptate pentru a ține tamponul plin. Cu un **server de cereri** (pull server), atâta timp cât este loc în tampon pentru încă un bloc, programul de redare continuă să trimită către server cereri pentru câte un bloc suplimentar. Scopul său este să țină tamponul cât mai plin posibil.

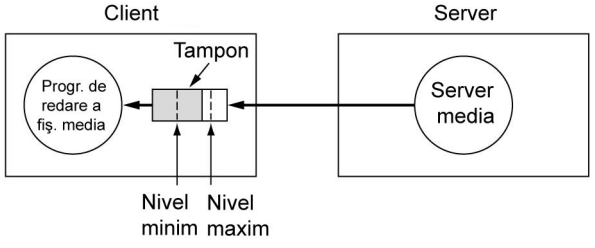


Fig. 7-61. Programul de redare a fișierelor media stochează intrarea de la serverul media și reproduce din tampon în loc să reproducă direct de pe rețea.

Dezavantajul unui server de cereri este existența unor cereri de date inutile. Serverul știe că a trimis tot fișierul, așa că de ce să punem programul de redare să întrebe încontinuu? Din acest motiv, soluția este folosită rar.

Cu un **server de forțare** (push server), programul de redare a fișierelor media trimite o cerere *PLAY*, iar serverul nu face decât să îi trimită date încontinuu. Aici sunt două posibilități: serverul media rulează cu viteză normală de playback sau rulează mai repede. În ambele cazuri, unele date sunt puse în tampon înainte de a începe playback-ul. Dacă serverul rulează la viteza normală de playback, datele care vin de la acesta sunt adăugate la sfârșitul tamponului, iar programul de redare șterge datele de la începutul tamponului. Atâta timp cât totul funcționează perfect, cantitatea de date din tampon rămâne constantă în timp. Această schemă este simplă, deoarece nu sunt necesare mesaje de control în nici o direcție.

Cealaltă schemă de forțare este cea în care serverul trimite date mai repede decât este nevoie. Avantajul acesteia este că dacă nu se poate garanta că serverul rulează cu o viteza constantă, acesta are ocazia să recupereze de fiecare dată când rămâne în urmă. O problemă este posibila depășire a cantității tamponului dacă serverul trimite date mai repede decât sunt consumate (și trebuie să poată face asta pentru a putea evita pauzele).

Soluția este ca programul de redare a fișierelor media să definească un **nivel minim** (low-water mark) și un **nivel maxim** (high-water mark) în tampon. Practic, serverul trimite date până când tamponul este umplut la nivelul maxim. Apoi programul de redare a fișierelor media îi spune să ia o pauză. Cum datele vor continua să intre în tampon până când serverul primește cererea de pauză, distanța de la nivelul maxim până la sfârșitul tamponului trebuie să fie mai mare decât întârzierea produsă de lățimea de bandă a rețelei. După ce serverul este oprit, tamponul începe să se golească. Când ajunge la nivelul minim, programul de redare a fișierelor media îi spune serverului media să reia trimiterea. Nivelul minim trebuie poziționat astfel încât tamponul să nu se golească integral.

Pentru a acționa asupra serverului de forțare, programul de redare a fișierelor media trebuie să îl controleze de la distanță. Acest lucru este asigurat de RTSP. Acesta este definit în RFC 2326 și oferă mecanisme de control al serverului pentru programul de redare. El nu se ocupă de fluxul de date, lucru făcut de obicei de RTP. Comenzile principale oferite de RTSP sunt listate în fig. 7-62.

Comanda	Răspunsul serverului
DESCRIBE	Afișează parametrii media
SETUP	Stabilește un canal logic între programul de redare și server.
PLAY	Începe să trimită date clientului.
RECORD	Începe să accepte date de la client.
PAUSE	Suspendă temporar trimiterea datelor.
TEARDOWN	Eliberează canalul logic.

Fig. 7-62. Comenzile RTSP de la programul de redare la server.

7.4.4 Radio prin Internet

O dată ce a devenit posibilă transmiterea de fluxuri audio prin Internet, stațiilor radio comerciale le-a venit ideea să emită și pe Internet, și prin aer. Nu cu mult timp după asta, stațiile radio ale universităților au început să își pună semnalul pe Internet. Apoi *studenții* și-au înființat propriile stații radio. Cu tehnologia actuală, practic oricine își poate înființa o stație radio. Întreaga zonă a radioului prin Internet este foarte nouă și în schimbare, însă merită să spunem câte ceva despre ea.

Există două soluții generale pentru radio-ul prin Internet. În prima, programele sunt pre-înregistrate și stocate pe disc. Ascultătorii se pot conecta la arhivele stației radio și pot alege și descărca orice program, pentru a-l asculta. De fapt, aceasta este similară cu fluxurile audio despre care tocmai am discutat. Este de asemenea posibil să se stocheze fiecare program imediat după ce a fost transmis în direct, astfel încât arhiva să funcționeze doar pentru, să zicem, o jumătate de oră sau mai puțin după transmisia în direct. Avantajele acestei soluții sunt că este ușor de realizat, toate tehnicile despre care am discutat funcționează și aici, iar ascultătorii pot alege dintre toate programele din arhivă.

Cealaltă soluție este transmiterea în direct pe Internet. Unele stații transmit prin aer și prin Internet simultan, dar sunt din ce în ce mai multe stații radio exclusiv pe Internet. Unele tehnici care sunt aplicabile fluxurilor audio sunt aplicabile și radio-ului în direct prin Internet, dar sunt și unele diferențe cheie.

Un element asemănător este necesitatea stocării într-un tampon în situl utilizatorului pentru a micșora fluctuațiile. Colectând 10 sau 15 secunde de radio înainte de începerea playback-ului, sunetul poate fi menținut continuu chiar și în cazul unor fluctuații substanțiale pe rețea. Atât timp cât pachetele ajung înainte să fie nevoie de ele, nu contează când au ajuns.

O diferență de bază este că fluxurile audio pot fi transmise cu o viteză mai mare decât viteza de playback, întrucât receptorul le poate opri când este atins nivelul maxim. Teoretic, asta îi dă timp pentru a retransmite pachetele pierdute, deși această abordare nu se folosește de obicei. În contrast, radio-ul în direct este întotdeauna transmis la exact aceeași rată cu care este generat și redat ascultătorului.

O altă diferență este că o stație radio în direct are de obicei sute sau mii de ascultători simultan, în timp ce fluxurile audio sunt punct la punct. În aceste condiții, radio-ul prin Internet ar trebui să folosească trimiterea multiplă cu protocoalele RTP/RTSP. Aceasta este cu siguranță cea mai eficientă cale de a funcționa.

În practica actuală, radio-ul prin Internet nu funcționează așa. Ceea ce se întâmplă de fapt este că utilizatorul stabilește o conexiune TCP cu stația și fluxul este trimis prin conexiunea TCP. Firește, asta creează diverse probleme, precum oprirea fluxului când fereastra este plină, pierderea pachetelor care au expirat și sunt retransmise, și așa mai departe.

Motivul pentru care se folosește trimiterea singulară TCP în locul trimiterii multiple RTP are trei părți. Prima, puține ISP-uri suportă trimiterea multiplă, astfel că aceasta nu este practic o opțiune. A doua, RTP este mai puțin cunoscut decât TCP și stațiile radio sunt adesea mici și au puține cunoștințe despre calculatoare, așa încât este mai ușor de folosit un protocol înțeles pe scară largă și suportat de toate pachetele de aplicații. A treia, mulți oameni ascultă radio-ul prin Internet la serviciu, ceea ce, în practică, înseamnă adesea în spatele unui zid de protecție (firewall). Mulți administratori de sistem își configurează zidul de protecție pentru a-și proteja LAN-ul de vizitatori nepoftiți. De obicei ei acceptă conexiuni TCP de la portul de la distanță 25 (SMTP pentru poștă electronică), pachete UDP de la portul de la distantă 53 (DNS), și conexiuni TCP de la portul de la distanță 80 (HTTP pentru Web). Aproape orice altceva poate fi blocat, inclusiv RTP. Astfel, singura cale de a obține semnalul radio prin zidul de protecție este ca situl Web să pretindă că este un server HTTP, cel puțin în fața zidului de protecție, și să folosească servere HTTP care utilizează TCP. Aceste măsuri severe, în timp ce oferă doar o securitate minimală, împing adesea cu forța aplicațiile multimedia către moduri de operare cu o eficiență drastic mai mică.

Cum radio-ul prin Internet este un mediu nou, războaiele asupra formatelor sunt în plină înflorire. Real Audio, Windows Media Audio și MP3 concurează agresiv pe această piață pentru a deveni formatul dominant în radio-ul prin Internet. Un nou venit este Vorbis, care este tehnic similar cu

MP3, însă are sursele disponibile și este suficient de diferit pentru a nu folosi patentele pe care se bazează MP3.

O stație tipică de radio prin Internet are o pagină Web ce îi listează programul, informații despre DJ și crainicii săi și multe reclame. Sunt de asemenea și una sau mai multe iconițe pentru a afișa formatele audio pe care le suportă (sau doar ASCULTĂ ACUM dacă un singur format este suportat). Aceste iconițe sau ASCULTĂ ACUM sunt metafișiere legate de tipul celor de care am discutat mai sus.

Când un utilizator selectează una dintre iconițe cu mouse-ul, este trimis metafișierul. Programul de navigare îi folosește tipul MIME sau extensia de fișier pentru a determina aplicația ajutătoare cea mai potrivită pentru metafișier (spre exemplu programul de redare a fișierelor media). Apoi scrie metafișierul într-un fișier auxiliar pe disc, pornește programul de redare a fișierelor media și îi trimite numele fișierului auxiliar. Programul de redare a fișierelor media citește fișierul auxiliar, vede URL-ul pe care acesta îl conține (de obicei o schemă *http* în loc de *rtsp* pentru a evita problema zidului de protecție și pentru că unele aplicații multimedia de succes lucrează astfel), contactează serverul și începe să funcționeze ca un radio. Ca element exterior, audio conține un singur flux, astfel că *http*-ul funcționează, însă pentru video, care are cel puțin două fluxuri, *http*-ul nu mai e bun și este necesar ceva de genul *rtsp*.

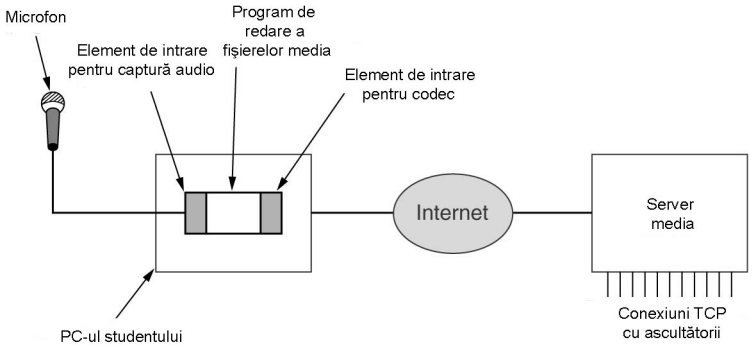


Fig. 7-63. O stație radio student.

O altă dezvoltare interesantă în zona radio-ului prin Internet este o organizare prin care oricine, chiar și un student, poate să înființeze și să opereze o stație radio. Componentele principale sunt ilustrate în fig. 7-63. Elementul de bază al stației este un calculator normal cu o placă de sunet și un microfon. Aplicațiile constau într-un program de redare a fișierelor media, precum Winamp sau Freeamp, cu un element de intrare pentru captură audio și un codec pentru formatul audio de ieșire selectat, spre exemplu MP3 sau Vorbis.

Fluxul audio generat de stație este apoi trimis prin Internet către un server mare, care se ocupă de distribuția acestuia unui număr mare de conexiuni TCP. De obicei serverul suportă multe stații mici. Acesta menține de asemenea un director cu stațiile pe care le are și ce emite fiecare în momentul curent. Potențialii ascultători merg la server, selectează o stație, și primesc date TCP. Există pachete de aplicații comerciale pentru controlul tuturor părților, precum și pachete cu sursele disponibile precum icecast. Există de asemenea și servere care sunt doritoare să se ocupe de distribuție contra unei taxe.

7.4.5 Voce peste IP

Cu mult timp în urmă, sistemul telefonic public comutat era în primul rând utilizat pentru traficul de voce cu variații mici de trafic de date. Dar traficul de date a crescut din ce în ce mai mult, încât în 1999, numărul de biți de date transportați era egal cu numărul de biți de voce (deoarece vocea este în PCM pe legăturile principale, poate fi măsurată în biți/sec). Până în 2002, volumul de trafic de date era cu un ordin de mărime mai mare decât volumul de trafic de voce și încă creștea exponențial, traficul de voce fiind aproape la același nivel (5% creștere pe an).

Ca o consecință a acestor numere, mulți operatori de rețele de pachete comutate au devenit dintr-o dată interesați de transportul vocii prin rețelele lor de date. Cantitatea suplimentară de bandă de transfer necesară pentru voce este minusculă, din moment ce rețelele de pachete au o dimensiune specifică traficului de date. Cu toate acestea, nota de plată a unei persoane este mai mare pentru telefon, decât pentru Internet, în felul acesta operatorii de rețea văzând telefonia pe Internet ca un mod de a câștiga mai mulți bani fără să mai pună alte cabluri în pământ. Astfel a luat naștere **telefonia pe Internet** (cunoscută și sub numele de **voce peste IP**).

H.323

Un lucru era clar pentru toată lumea încă de la început și anume că dacă fiecare vânzător și-ar fi creat propria stivă de protocoale, sistemul nu ar fi funcționat niciodată. Pentru a evita această problemă, un număr de participanți interesați s-au adunat sub auspiciile ITU pentru a realiza standardele. În 1996 ITU a lansat o recomandare **H.323** intitulată "Sisteme de telefonie vizuală și echipamente pentru rețele locale care nu garantează calitatea serviciului". Doar industria telefonică ar putea gândi un astfel de nume. Recomandarea a fost revizuită în 1998 și apoi acest H.323 revizuit a fost baza primelor sisteme universale de telefonie pe Internet.

H.323 este mai mult o prezentare arhitecturală a telefoniei pe Internet decât un protocol specific. El se referă la un număr mare de protocoale specifice pentru codificarea vocii, configurarea apelului, semnalizare, transportul datelor și alte aspecte mai mult decât să specifice el însuși aceste lucruri. Modelul general este reprezentat în fig. 7-64. În centru este o **poartă (gateway)** care conectează la Internet rețeaua de telefonie. Comunică prin protocoalele H.323 pe partea de Internet și prin protocoalele PSTN pe partea de telefonie. Dispozitivele de comunicație sunt denumite **terminale**. Un LAN poate avea un **administrator de poartă (gatekeeper)**, care controlează punctele finale de sub jurisdicția sa, numite **zone**.

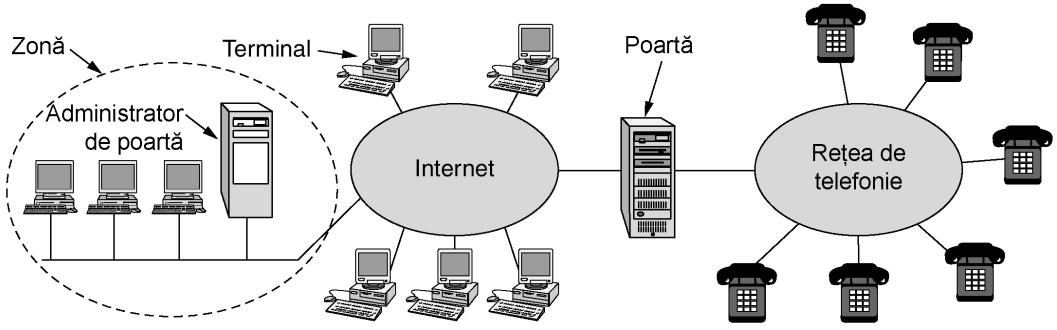


Fig. 7-64. Modelul arhitectural H.323 pentru telefonia prin Internet.

O rețea de telefonie are nevoie de un număr de protocoale. Să începem cu protocolul de codificare și decodificare a vocii. Sistemul PCM pe care l-am studiat în Cap. 2 este definit în recomandarea ITU **G.711**. Acesta codifică un singur canal de voce prin eșantionări de 8000 de ori pe secundă cu un model pe 8 biți care să redea vocea necomprimată la 64 Kbps. Toate sistemele H.323 trebuie să suporte G.711. Cu toate acestea, alte protocoale de compresie a vocii sunt de asemenea permise (dar nu obligatorii). Acestea folosesc diferiți algoritmi de compresie pentru a face diferite compromisuri între calitate și lărgime de bandă. De exemplu, **G.723.1** preia un bloc de 240 de eșantioane (30 ms de voce) și folosește codificarea predictivă pentru a-l reduce la 24 octeți sau 20 octeți. Acest algoritm oferă o rată la ieșire de 6,4 Kbps sau 5,3 Kbps (factori de compresie de 10 și 12), respectiv, cu mici pierderi în calitatea percepută. Sunt, de asemenea, permise și alte codificări.

Din moment ce sunt permiși mai mulți algoritmi de compresie, este necesar un protocol care să permită terminalelor să negocieze ce algoritm vor folosi. Acest protocol poartă numele de **H.245**. De asemenea, se negociază și alte aspecte ale conexiunii, ca de exemplu viteza de transmisie. RTCP este necesar pentru controlul canalelor RTP. De asemenea, este necesar un protocol pentru stabilirea și eliberarea conexiunilor, asigurarea de tonuri, crearea de sunete de apel și restul telefoniei standard. Aici este utilizat ITU **Q.931**. Terminalele au nevoie de un protocol pentru a comunica cu administratorul de poartă (când acesta există). În acest scop este folosit **H.255**. Canalul PC – administrator de poartă pe care îl gestionează este denumit canal **RAS (Registration/Admission/Status, rom: Înregistrare/Admisie/Stare)**. Acest canal permite terminalelor să intre sau să părăsească zona, să ceară sau să elibereze bandă de transfer și să asigure, printre altele, actualizări de stare. În fine, un protocol este necesar pentru transmiterea efectivă a datelor. RTP este utilizat în acest scop. Este administrat de RTCP, ca de obicei. Poziționarea tuturor protocoalelor este prezentată în fig. 7-65.

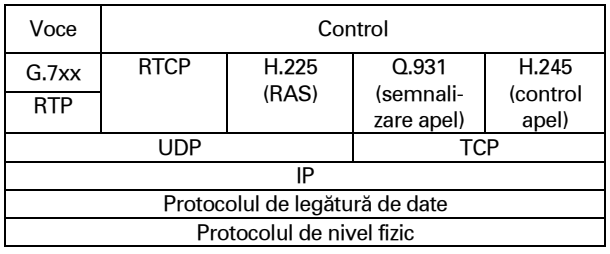


Fig. 7-65. Stiva de protocoale H.323.

Pentru a vedea cum lucrează împreună aceste protocoale, să considerăm cazul unui terminal PC pe un LAN (cu un administrator de poartă) care apelează un telefon aflat la distanță. Mai întâi, PC-ul trebuie să localizeze administratorul de poartă, astfel că difuzează un pachet UDP de aflare a administratorului de poartă pe portul 1718. Când administratorul de poartă răspunde, PC-ul află adresa IP a administratorului de poartă. Acum PC-ul se înregistrează la administratorul de poartă trimițându-i un mesaj RAS într-un pachet UDP. După ce a fost acceptat, PC-ul trimite administratorului de poartă un mesaj de admitere RAS, cerând lărgime de bandă. Numai după ce banda a fost acordată se poate face inițierea apelului. Ideea de a cere lărgime de bandă în avans este aceea de a permite administratorului de poartă să limiteze numărul de apeluri pentru a evita supraîncărcarea liniei de ieșire și a ajuta la asigurarea calității necesare a serviciului.

În acest moment, PC-ul stabilește o conexiune TCP cu administratorul de poartă pentru a iniția apelul. Configurarea apelului folosește protocoale existente de rețea de telefonie, care sunt orientate pe conexiuni, deci este necesar TCP-ul. În contrast, sistemul telefonic nu are nimic echivalent

canalului RAS pentru a permite telefoanelor să-și anunțe prezența, astfel creatorii H.323 au fost nevoiți să folosească fie UDP, fie TCP pentru RAS, și au ales protocolul cu supraîncărcarea cea mai mică, UDP.

În acest moment, când PC-ul are alocată bandă de transfer, el poate să trimită un mesaj Q.931 *SETUP* (configurare) peste conexiunea TCP. Acest mesaj specifică numărul de telefon apelat (sau adresa IP și portul, dacă este apelat un calculator). Administratorul de poartă răspunde cu un mesaj Q.931 *CALL PROCEEDING* (începerea comunicării) pentru a confirma primirea cererii. Administratorul de poartă trimite mai departe mesajul *SETUP* către poartă.

Poarta, care este jumătate calculator, jumătate comutator telefonic, lansează un apel obișnuit către telefonul dorit. Oficiul final la care este legat telefonul, apelează telefonul destinație și trimite de asemenea un mesaj Q.931 *ALERT* (alertă) pentru a anunța PC-ul apelant că a început să sune. Când persoana de la celălalt capăt al firului ridică receptorul, oficiul final trimite înapoi un mesaj Q.931 *CONNECT* (conectare) pentru a anunța PC-ul că are o conexiune.

Odată stabilită conexiunea, administratorul de poartă nu mai este în buclă, deși poarta încă mai este. Pachetele următoare trec peste administratorul de poartă, ducându-se direct la adresa IP a porții. În acest punct, avem doar un simplu tub între cele două părți. Aceasta este doar o conexiune la nivel fizic pentru transferul biților, nimic mai mult. Nici un capăt nu știe nimic despre celălalt.

Protocolul H.245 este acum folosit pentru negocierea parametrilor apelului. El folosește canalul de control H.245, care este întotdeauna deschis. Fiecare parte începe prin anunțarea capabilităților sale, de exemplu, dacă poate suporta apeluri video (H.323 suportă apeluri video) sau conferințe, ce codificări suportă etc. În momentul în care fiecare capăt știe ce suportă celălalt, sunt stabilite două canale unidirecționale și un codor și alți parametri sunt atribuiți fiecăruia. Din moment ce fiecare capăt poate avea un echipament diferit, este foarte probabil să fie diferite și codoarele pe canalele de trimitere și recepție. După ce s-au încheiat toate negocierile, fluxul de date utilizând RTP poate începe. El este administrat prin RTCP, care joacă un rol în controlul congestiei. Dacă sunt prezente transmisii video, RTCP se ocupă de sincronizarea audio/video. Diferitele canale sunt ilustrate în fig. 7-66. Când oricare din cele două capete se închide, canalul Q.931 de semnalizare al apelului este utilizat pentru a opri conexiunea.

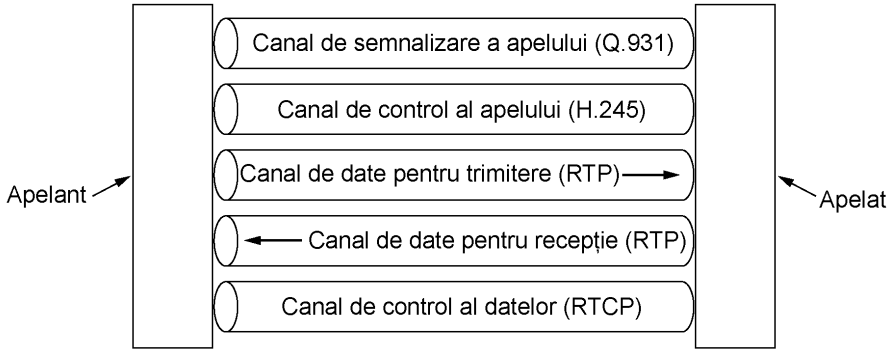

Canal de semnalizare a apelului (Q.931)
Canal de control al apelului (H.245)
Canal de date pentru trimitere (RTP)
Canal de date pentru recepție (RTP)
Canal de control al datelor (RTCP)
Apelant
Apelat

Fig. 7-66. Canale logice între apelant și apelat în timpul apelului.

Când se încheie apelul, PC-ul apelant contactează din nou administratorul de poartă cu un mesaj RAS pentru a elibera banda de transfer care i-a fost atribuită. Alternativ, el poate să facă un nou apel.

Nu am spus nimic despre calitatea serviciului, deși aceasta este esențială pentru a face din vocea peste IP un succes. Motivul este simplu, calitatea serviciului este în afara domeniului H.323. Dacă

rețeaua transportatoare este capabilă să producă o conexiune stabilă, fără distorsiuni dinspre PC-ul apelant (de exemplu, folosind tehnicile pe care le-am discutat în cap. 5) înspre poartă, atunci calitatea serviciului pe apel va fi bună; altfel, nu va fi. Partea telefonică folosește PCM și este întotdeauna fără distorsiuni.

SIP – Protocolul de inițiere a sesiunii

H.323 a fost conceput de ITU. Mulți oameni din comunitatea Internet l-au văzut ca un produs tipic telco: mare, complex, și inflexibil. În consecință, IETF a creat un comitet pentru a concepe un mod mai simplu și mai modular pentru vocea peste IP. Rezultatele cele mai bune până în prezent se concretizează în **SIP (Session Initiation Protocol, rom: Protocolul de inițiere a sesiunii)**, care este descris în RFC 3261. Protocolul descrie configurarea apelurilor telefonice pe Internet, video conferințele și alte conexiuni multimedia. Spre deosebire de H.323, care este o întreagă suită de protocoale, SIP este un singur modul, dar a fost conceput pentru a conlucra bine cu aplicațiile Internet existente. De exemplu, definește numerele de telefon ca URL-uri, pentru a fi incluse în pagini Web, permițând ca un clic pe o legătură să inițieze un apel telefonic (asemănător, schema *mailto* permite ca activarea unei hiper-legături să deschidă programul de trimitere al unui mesaj electronic).

SIP poate stabili sesiuni bilaterale (apeluri telefonice obișnuite), sesiuni multilaterale (în care oricine poate auzi și vorbi), și sesiuni cu transmisie multiplă (un emițător, mai mulți receptori). Sesiunile pot conține audio, video, sau date, ultimul fiind folositor de exemplu pentru jocuri cu mai mulți utilizatori în timp real. SIP se ocupă doar cu configurarea, administrarea și terminarea sesiunilor. Alte protocoale, ca RTP/RTCP, sunt utilizate pentru transportul datelor. SIP este un protocol de nivel aplicație și poate rula peste UDP sau TCP.

SIP suportă o varietate de servicii, inclusiv localizarea apelatului (care poate nu este la calculatorul său de acasă) și să determine capabilitățile acestuia, precum și să trateze mecanismele de configurare și terminare. În cel mai simplu caz, SIP setează o sesiune de la calculatorul apelantului la calculatorul apelatului, deci să tratăm mai întâi acest caz.

Numerele de telefon în SIP sunt reprezentate ca URL-uri utilizând schema *sip*, de exemplu, *sip:ilse@cs.university.edu* pentru utilizatorul ilse de pe calculatorul specificat de numele de DNS *cs.university.edu*. URL-urile SIP pot conține adrese IPv4, IPv6, sau chiar numere de telefon.

Protocolul SIP este un protocol bazat pe text modelat în HTTP. Un capăt trimite un mesaj în text ASCII ce conține un nume de metodă pe prima linie, urmată de linii adiționale ce conțin antete pentru transmiterea parametrilor. Multe dintre antete sunt luate din MIME pentru a permite protocolului SIP să conlucreze cu aplicațiile Internet existente. Cele șase metode definite de specificația de bază sunt enumerate în fig. 7-67.

Metoda	Descriere
INVITE	Cerere de inițiere a unei sesiuni
ACK	Confirmare că o sesiune a fost inițiată
BYE	Cerere de terminare a unei sesiuni
OPTIONS	Interogarea unui calculator despre capabilitățile sale
CANCEL	Anularea unei cereri în așteptare
REGISTER	Informarea unui server de redirecționare despre locația curentă a utilizatorului

Fig. 7-67. Metodele SIP definite în specificația de bază.

Pentru stabilirea unei sesiuni, apelantul fie creează o conexiune TCP cu apelatul și trimite un mesaj *INVITE*, fie trimite mesajul *INVITE* într-un pachet UDP. În ambele cazuri, antetele din a

doua și următoarele linii descriu structura corpului mesajului ce conțin capabilitățile apelantului, tipurile de mediu de transmisie și formatele. Dacă cel apelat acceptă convorbirea, el răspunde cu un cod de răspuns de tip HTTP (un număr de trei digiți conform grupurilor din fig. 7-42, 200 pentru acceptare). După linia cu codul de răspuns, apelatul poate de asemenea include informații despre capabilitățile sale, tipurile de mediu de transmisie și formate.

Conexiunea este realizată prin mecanismul înțelegerii în trei pași astfel că, pentru a încheia protocolul, apelantul răspunde cu un mesaj de confirmare a recepționării mesajului 200.

Oricare dintre cele două capete pot cere terminarea sesiunii prin trimiterea unui mesaj conținând metoda *BYE*. Când celălalt capăt trimite confirmarea primirii acestuia, sesiunea este terminată.

Metoda *OPTIONS* este utilizată pentru a interoga o mașină despre propriile sale capabilități. Este în general folosită înainte de inițierea unei sesiuni pentru a afla dacă acea mașină este capabilă de transmisii de voce peste IP sau ce alt tip de sesiune este urmărit.

Metoda *REGISTER* se referă la abilitatea protocolului SIP de a urmări și a se conecta la un utilizator care nu este acasă. Acest mesaj este trimis unui server de localizare SIP care ține o evidență a locațiilor utilizatorilor. Acel server poate fi mai târziu interogat pentru a afla locația curentă a utilizatorului. Operația de redirecționare este ilustrată în fig. 7-68. Aici, apelantul trimite un mesaj *INVITE* unui server proxy pentru a ascunde posibila redirecționare. Proxy-ul caută apoi unde este utilizatorul și îi trimite un mesaj *INVITE*. Apoi se comportă ca un intermediar pentru mesajele următoare în înțelegerea în trei pași. Mesajele *LOOKUP* și *REPLY* nu fac parte din SIP; orice protocol convenabil poate fi utilizat, depinzând de tipul de server de localizare utilizat.

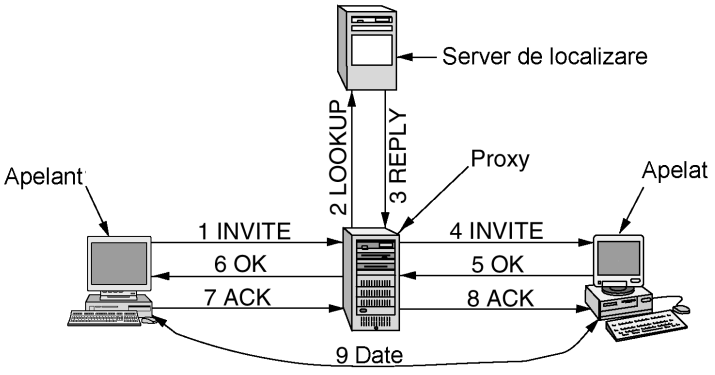


Fig. 7-68. Utilizarea unui proxy și servere de redirecționare cu SIP.

SIP are o varietate de alte caracteristici pe care noi nu le vom descrie aici, incluzând așteptarea apelului, ecranarea apelului, criptare și autentificare. De asemenea, are abilitatea de a apela de la un calculator, un telefon obișnuit, dacă este disponibilă o poartă de conversie corespunzătoare între Internet și sistemul telefonic.

Comparație între H.323 și SIP

H.323 și SIP au multe puncte în comun, dar și diferențe. Ambele protocoale permit apeluri bilaterale și multilaterale folosind atât calculatoare, cât și telefoane ca puncte finale. Ambele suportă negocierea parametrilor, criptarea și protocoalele RTP/RTCP. Un rezumat al acestor asemănări și diferențe este dat în fig. 7-69.

Noțiune	H.323	SIP
Conceput de	ITU	IETF
Compatibilitate cu PSTN	Da	În mare parte
Compatibilitate cu Internet	Nu	Da
Arhitectura	Monolitică	Modulară
Completitudine	Întreaga stivă de protocoale	SIP tratează doar configurarea
Negocierea parametrilor	Da	Da
Semnalizarea apelului	Q.931 peste TCP	SIP peste TCP și UDP
Formatul mesajului	Binar	ASCII
Mediul de transmisie	RTP/RTCP	RTP/RTCP
Apeluri multilaterale	Da	Da
Conferințe multimedia	Da	Nu
Adresare	Calculator sau număr de telefon	URL
Terminarea apelului	Explicită sau eliberare TCP	Explicită sau la expirarea timpului
Mesagerie imediată	Nu	Da
Criptare	Da	Da
Dimensiunea standardelor	1400 pagini	250 pagini
Implementare	Mare și complexă	Moderată
Stare	Larg răspândită	Prezentă și viitoare

Fig. 7-69. Comparație între H.323 și SIP

Deși seturile de caracteristici sunt similare, cele două protocoale diferă mult în concepție. H.323 este un standard tipic, cu greutate, al industriei de telefonie, specificând întreaga stivă de protocoale și definind exact ce este permis și ce este interzis. Abordarea duce la protocoale bine definite la fiecare nivel, ușurând interoperabilitatea. Prețul este un standard mare, complex și rigid, dificil de adaptat la aplicațiile viitoare.

În contrast, SIP este un protocol tipic de Internet care lucrează prin schimbul de linii scurte de text ASCII. Este un modul ușor care conlucrează bine cu alte protocoale de Internet, dar mai puțin cu protocoalele de semnalizare din sistemul telefonic existent. Deoarece modelul IETF al vocii peste IP este în mare măsură modular, el este flexibil și poate fi adaptat cu ușurință la noi aplicații. Partea neplăcută este cea a potențialelor probleme de interoperabilitate, deși acestea sunt discutate în întâlniri frecvente unde diverși implementatori își testează împreună sistemele.

Vocea peste IP este o temă prezentă și viitoare. De aceea, deja există cărți pe această temă. Câteva exemple sunt (Collins, 2001; Davidson și Peters, 2000; Kumar ș.a.., 2001; și Wright, 2001). Ediția din mai/iunie 2002 a revistei *Internet Computing* are mai multe articole pe această temă.

7.4.6 Introducere la video

Până acum am discutat urechea în detaliu; este timpul să ne mutăm la ochi (nu, această secțiune nu este urmată de una despre nas). Ochiul uman are proprietatea că atunci când o imagine apare pe retină, imaginea este păstrată acolo pentru câteva milisecunde. Dacă o secvență de imagini este desenată linie cu linie la 50 imagini/sec, ochiul nu observă că privește imagini discrete. Toate sistemele video (de exemplu, televizorul) exploatează acest principiu pentru a produce imagini în mișcare.

Sisteme Analogice

Pentru a înțelege sistemele video, este bine să pornim de la vechea televiziune simplă, alb-negru. Pentru a reprezenta imaginile bidimensionale din fața ei ca o tensiune unidimensională funcție de timp, camera de luat vederi scanează imaginea cu o rază electronică, rapid de-a latul și lent în josul

ei, înregistrând intensitatea luminoasă așa cum vine. La sfârșitul scanării, numit **cadru** (frame), raza reia traseul. Această intensitate este difuzată ca funcție de timp, iar receptorii repetă procesul de scanare pentru reconstrucția imaginii. Modelul de scanare folosit atât de cameră cât și de receptor, este prezentat în fig. 7-70. (Camerele CCD integrează mai degrabă decât scanează, dar unele camere și toate monitoarele scanează.)

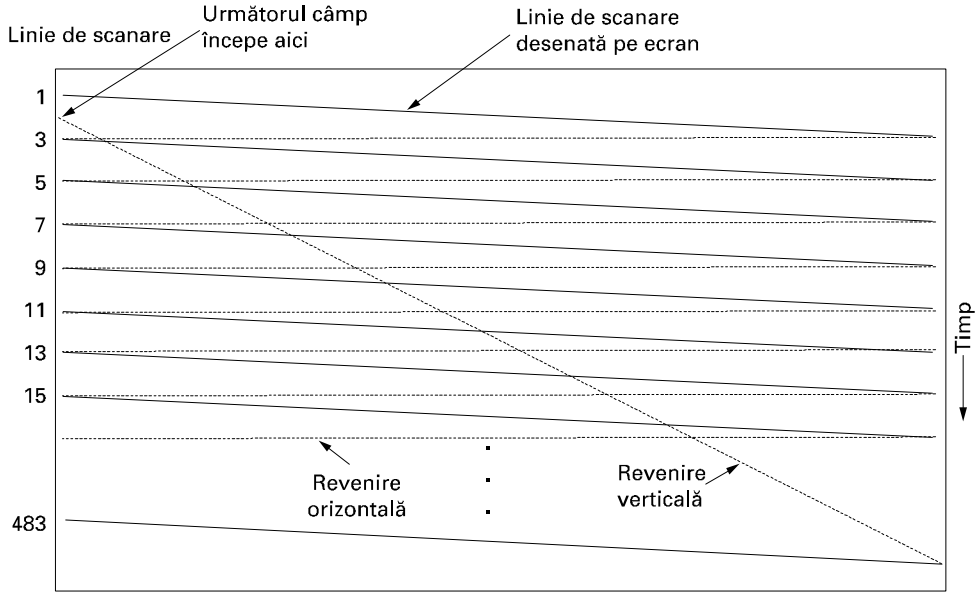


Fig. 7-70. Modelul de scanare folosit pentru video și televiziunea NTSC.

Parametrii exacți de scanare variază de la țară la țară. Sistemul folosit în America de Nord și de Sud și Japonia are 525 de linii de scanare, o rată de aspect orizontal/vertical de 4:3 și 30 de cadre/sec. Sistemul european are 625 de linii de scanare, aceeași rată de aspect de 4:3 și 25 de cadre/sec. În ambele sisteme, câteva linii din vârf și câteva din partea de jos nu sunt afișate (pentru a aproxima o imagine dreptunghiulară pe un tub catodic rotund original). Doar 483 din cele 525 de linii de scanare NTSC (și 576 din cele 625 de linii de scanare PAL/SECAM) sunt afișate. Raza este stinsă în timpul revenirii verticale, așa că multe stații (în special în Europa) folosesc acest interval pentru difuzare de TeleText (pagini de text conținând știri, vreme, sporturi, prețuri la bursă etc.).

În timp ce 25 de cadre/sec sunt suficiente pentru a capta o mișcare lină, la această viteză a cadrelor multe persoane, în special cei bătrâni, vor percepe imaginea tremurată (deoarece imaginea veche a fost ștearsă de pe retină înaintea apariției uneia noi). În loc să se mărească viteza cadrelor, care va cere să se folosească mai puțină lărgime de bandă, este aleasă o altă cale. În locul afișării liniilor de scanare în ordine, întâi sunt afișate toate liniile de scanare cu numere impare, apoi cele cu numere pare. Fiecare din aceste jumătăți de cadre este numită **câmp (field)**. Experimentele arată că deși oamenii remarcă o pâlpâire la 25 de cadre/sec, ei nu o remarcă la 50 de cadre/sec. Această tehnică este numită **întrețesere (interlacing)**. O televiziune sau video neîntrețesută este denumită **progresivă (progressive)**. Observați că filmele rulează la 24 fps, dar fiecare cadru este vizibil în totalitate pentru 1/24 sec.

Video-ul color folosește același model de scanare ca monocromul (alb și negru), cu excepția faptului că în locul afișării unei imagini cu o singură rază mișcătoare, sunt folosite trei raze care se mișcă

la unison. Este folosită câte o rază pentru fiecare din cele trei culori aditive primare: roșu, verde și albastru (RGB). Această tehnică funcționează pentru că orice culoare poate fi construită din superpoziția liniară a roșului, verdelui și albastrului cu intensitățile corespunzătoare. Cu toate acestea, pentru transmiterea pe un singur canal, cele trei semnale de culori trebuie combinate într-un singur semnal **compus (composite)**.

Atunci când a apărut televiziunea color, diverse metode de afișare color erau tehnic posibile, și diverse țări au făcut alegeri diferite, conducând la sisteme care sunt încă incompatibile. (Observați că aceste alegeri nu au nimic în comun cu VHS, față de Betamax și față de P2000, care sunt tehnici de înregistrare.) În toate țările, o necesitate politică a fost ca programele color să fie recepționate de televizoarele existente alb-negru. În consecință, cea mai simplă schemă, care codifică separat semnalele RGB, nu a fost acceptată. De asemenea, RGB nu este cea mai eficientă schemă.

Primul sistem color a fost standardizat în Statele Unite de **Comitetul Național de Standarde de Televiziune (National Television Standards Committee)**, care a împrumutat acronimul său standardului: **NTSC**. Televiziunea color a fost introdusă în Europa câțiva ani mai târziu, în momentul în care tehnologia a fost îmbunătățită substanțial, conducând la sisteme cu mare imunitate la zgomote și culori mai bune. Acestea sunt numite **SECAM (SEquentiel Couleur Avec Memoire, rom: culoare secvențială cu memorie),** sistem folosit în Franța și în țările din estul Europei, și **PAL (Phase Alternating Line, rom: linie cu fază alternată)** folosit în restul Europei. Diferența în calitatea culorii între NTSC și PAL/SECAM a condus la gluma că NTSC înseamnă de fapt că aceeași culoare nu apare de două ori (Never Twice the Same Color).

Pentru a permite ca transmisiunile color să fie văzute de receptoarele alb-negru, toate cele trei sisteme combină liniar semnalele RGB într-un semnal de **luminanță (luminance)**, și două semnale de **crominanță (chrominance)**, deși toate folosesc alți coeficienți pentru construcția acestor semnale din semnale RGB. Interesant, ochiul este mai sensibil la semnalele de luminanță decât la cele de crominanță, astfel încât ultimele nu trebuie transmise cu mare acuratețe. Ca rezultat, semnalul luminos poate fi difuzat la aceeași frecvență ca și vechiul semnal alb-negru, așa încât poate fi recepționat pe televizoarele existente alb-negru. Cele două semnale de crominanță sunt difuzate în benzi înguste la frecvențe înalte. Unele televizoare au butoane pentru controlul strălucirii, nuanței și saturării (sau strălucirii, tentei și culorii) pentru controlul separat al celor trei semnale. Înțelegerea luminanței și crominanței este vitală pentru înțelegerea modului în care funcționează compresia video.

În ultimii ani, a existat un interes considerabil pentru **HDTV (High Definition TeleVision, rom: televiziunea de înaltă definiție)**, care produce imagini mai bune dublând numărul liniilor de scanare. Statele Unite, Europa și Japonia au dezvoltat sisteme HDTV, toate diferite și toate mutual incompatibile. V-ați fi așteptat la altceva? Principiile de bază ale HDTV-ului în termeni de scanare, luminanță, crominanță și altele, sunt similare sistemelor existente. Cu toate acestea, toate cele trei formate au o aceeași rată a aspectului de 16:9 în loc de 4:3 pentru a le potrivi mai bine formatului folosit pentru filme (care sunt înregistrate pe 35 mm, cu o rată de aspect de 3:2).

Sisteme digitale

Cea mai simplă reprezentare a video-ului digital este o secvență de cadre, fiecare constând dintr-o grilă dreptunghiulară de elemente de imagine, adică **pixeli**. Fiecare pixel poate fi un singur bit, pentru a reprezenta fie albul, fie negrul. Calitatea unui astfel de sistem este similară cu cea obținută la transmiterea prin fax a unei fotografii color - adică groaznică. (Încercați dacă puteți; sau fotocopiați o fotografie color la o mașină de copiat care nu rasterizează.)

Următorul pas este de a folosi 8 biți pe pixel pentru a reprezenta 256 nivele de gri. Această schemă dă o calitate ridicată video-ului alb-negru. Pentru video color, sistemele bune folosesc 8 biți pentru fiecare din culorile RGB, deși aproape toate sistemele le amestecă pentru transmitere într-un video compus. În timp ce folosind 24 de biți per pixel se limitează numărul de culori la 16 milioane, ochiul uman nu poate deosebi atâtea culori, deci nici vorbă de mai multe. Imaginile digitale color sunt produse folosind trei raze de scanare, una pentru fiecare culoare. Geometria este aceeași cu cea pentru sistemul analogic din fig. 7-70, exceptând faptul că liniile de scanare continue sunt înlocuite acum de linii formate din pixeli discreți.

Pentru a produce mișcări line, video-ul digital, la fel ca video-ul analog, trebuie să afișeze cel puțin 25 de cadre/sec. Oricum, deoarece monitoarele de bună calitate ale calculatoarelor rescanează deseori ecranul din imagini memorate de 75 de ori pe secundă sau mai des, întrețeserea nu este necesară și, în consecință, nu este folosită în mod obișnuit. Reafișarea (adică redesenarea) aceluiași cadru de trei ori la rând este suficientă pentru eliminarea pâlpâirii.

Cu alte cuvinte, continuitatea unei mișcări este determinată de numărul de imagini *diferite* pe secundă, având în vedere că pâlpâirea este determinată de numărul de ori pe secundă în care este desenat ecranul. Acești doi parametri sunt diferiți. O imagine afișată la 20 de cadre/sec nu va arăta distorsionată, dar va pâlpâi, deoarece un cadru va dispărea de pe retină înainte ca următorul cadru să apară. Un film cu 20 de cadre diferite pe secundă, fiecare dintre acestea fiind desenat de patru ori la rând, nu va pâlpâi, dar va părea distorsionat.

Semnificația acestor doi parametri devine mai clară atunci când considerăm lărgimea de bandă pentru transmisia video digitală printr-o rețea. Cele mai multe din monitoarele actuale folosesc rata de aspect de 4:3, astfel încât pot folosi tuburile ieftine din producția destinată pieței televizoarelor. Configurațiile obișnuite sunt 1024x768, 1280x960, și 1600x1200. Chiar și cel mai mic dintre acestea cu 24 biți pe pixel și 25 cadre/sec are nevoie să fie alimentat la 472 Mbps. Pentru aceasta ar fi nevoie de un purtător SONET OC-12, și de altfel rularea unui purtător OC-9 SONET în casa fiecăruia nu este chiar la ordinea zilei. Dublarea acestei rate pentru a evita pâlpâirea este chiar mai puțin de dorit. O soluție mai bună este transmiterea a 25 cadre/sec pe care calculatorul să le memoreze și să le afișeze de 2 ori. Televiziunea obișnuită nu folosește această strategie, deoarece televizoarele nu au memorie. Și chiar dacă ar avea memorie, sistemele analogice nu pot fi memorate în RAM fără o conversie prealabilă în formă digitală, care necesită hardware în plus. Ca o consecință, întrețeserea este necesară pentru televiziunea obișnuită, dar nu și pentru video digital.

7.4.7 Compresia video

Ar trebui să fie evident acum că transmisia materialului video în formă necomprimată nu intră în discuție. Singura speranță este într-o compresie puternică. Din fericire, numeroase cercetări de câteva decenii au ajuns la mai multe tehnici de compresie și algoritmi care fac posibilă transmisia de multimedia. În această secțiune vom studia cum este realizată compresia video.

Toate sistemele de compresie necesită doi algoritmi: unul pentru comprimarea datelor la sursă și altul pentru decomprimarea lor la destinație. În literatură, acești algoritmi sunt denumiți algoritmi de **codificare** și **decodificare**. Vom folosi și aici această terminologie.

Acești algoritmi prezintă câteva asimetrii a căror înțelegere este importantă. Mai întâi, pentru multe aplicații, un document multimedia, să zicem un film, va fi codificat o dată (atunci când este memorat pe un server multimedia), dar va fi decodificat de milioane de ori (atunci când este vizualizat de clienți). Această asimetrie înseamnă că este acceptabil ca algoritmul de codificare să fie lent și

să necesite un hardware suplimentar scump, cu condiția ca algoritmul de decodificare să fie rapid și să nu ceară un hardware costisitor. Mai mult, operatorul unui server multimedia ar putea dori să închirieze un supercalculator paralel pentru câteva săptămâni, pentru a-și codifica întreaga videotecă, dar a cere clienților să închirieze un supercalculator pentru 2 ore, pentru a vedea un film nu va avea mare succes. Multe sisteme de compresie fac eforturi pentru ca decodificarea să fie rapidă și simplă, chiar cu prețul încetinirii și complicării codificării.

Pe de altă parte, pentru multimedia în timp-real, precum conferințe video, codificarea lentă este inacceptabilă. Codificarea trebuie făcută din mers, în timp-real. În consecință, multimedia de timp-real folosește algoritmi sau parametri diferiți față de cei utilizați pentru memorarea video-urilor pe disc, deseori cu mult mai puțină compresie.

A doua asimetrie este că procesul de codificare/decodificare nu trebuie să fie inversabil. Adică, atunci când se comprimă un fișier, acesta se transmite și apoi este decomprimat, utilizatorul așteptând să-l obțină pe cel original, exact până la ultimul bit. Cu multimedia, această cerință nu există. De obicei, este acceptabil ca un semnal video, după codificare și decodificare, să fie un pic diferit de original. Atunci când ieșirea decodificată nu este egală exact cu intrarea originală, sistemul se spune că este **cu pierderi** (lossy). Dacă intrarea și ieșirea sunt identice, sistemul este **fără pierderi** (lossless). Sistemele cu pierderi sunt importante, deoarece acceptarea unui număr mic de informații pierdute poate oferi un avantaj imens în termenii de rată de compresie posibilă.

Standardul JPEG

Un film este doar o secvență de imagini (plus sunet). Dacă am putea găsi un algoritm bun pentru codificarea unei singure imagini, acest algoritm ar putea fi aplicat succesiv fiecărei imagini, pentru a obține compresia video. Există algoritmi buni de compresie a imaginii, deci să începem acolo studiul nostru despre compresia video. Standardul **JPEG (Joint Photographic Experts Group, rom: grupul comun al experților fotografi)** pentru comprimarea imaginilor cu tonuri continue (de exemplu, fotografii), a fost dezvoltat de experții în fotografii lucrând sub auspiciile ITU, ISO și IEC, un alt organism de standarde. Este important pentru multimedia deoarece la o primă aproximare, standardul multimedia pentru filme, MPEG, este codificarea JPEG a fiecărui cadru separat, plus câteva caracteristici pentru comprimarea între cadre și detectarea mișcării. JPEG este definit în Standardul International 10918.

JPEG are patru moduri și multe opțiuni. Standardul este mai asemănător cu o listă de cumpărături decât cu un simplu algoritm. Pentru scopurile noastre, doar modul secvențial cu pierderi este relevant, iar acesta este ilustrat în fig. 7-71. Cu toate acestea, ne vom concentra asupra modului în care JPEG este folosit în mod normal pentru codificarea imaginilor video de 24-biți RGB și vom lăsa la o parte detalii minore pentru simplitate.

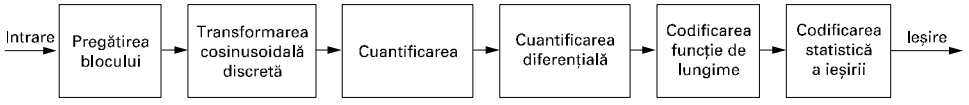


Fig. 7-71. Funcționarea JPEG în modul secvențial cu pierderi.

Pasul 1 de codificare a imaginii cu JPEG este pregătirea blocului. Pentru specificare, să presupunem că intrarea JPEG este o imagine RGB de 640x480 cu 24 biți/pixel, ca în fig. 7-72(a). Deoarece folosirea luminanței și crominanței dă o mai bună compresie, vom calcula mai întâi luminanța, Y, și cele două crominanțe, I și Q (pentru NTSC), după formulele următoare:

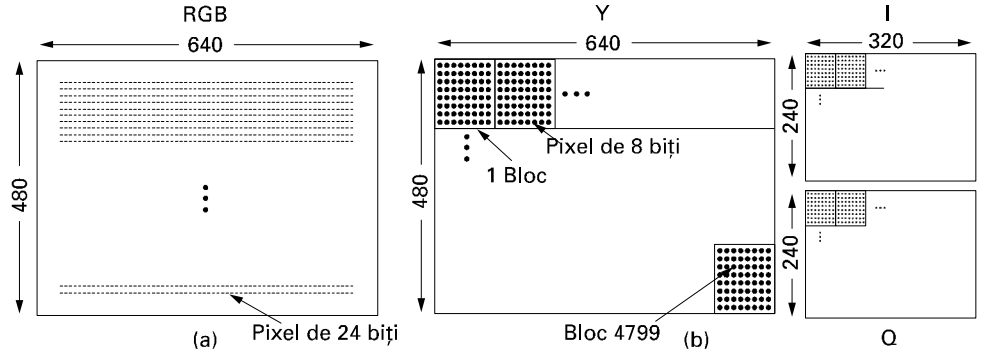


Fig. 7-72. (a) Date de intrare RGB. (b) După pregătirea blocului.

Y = 0,30R + 0,59G + 0,11B
I = 0,60R - 0,28G - 0,32B
Q = 0,21R - 0,52G + 0,31B

Pentru PAL, crominanțele sunt notate cu *U* și *V* și coeficienții diferă, dar ideea este aceeași. SECAM diferă atât de NTSC cât și de PAL.

Pentru *Y*,*I* și *Q* sunt construite matrice separate, fiecare cu elemente între 0 și 255. Apoi, blocuri pătrate de patru pixeli sunt aranjate în matricele *I* și *Q* pentru a le reduce la 320x240. Această reducere este cu pierderi, dar ochiul abia dacă observă, deoarece el răspunde la luminanță mai mult decât la crominanță. Cu toate acestea, întreaga cantitate de date este comprimată cu un factor de doi. Acum se scade 128 din fiecare element al celor trei matrice pentru a aduce zero la mijlocul intervalului. În fine, fiecare matrice este divizată în blocuri de 8x8. Matricea *Y* are 4800 blocuri; celelalte două au 1200 de blocuri fiecare, așa cum se arată în fig. 7-72(b).

Pasul 2 al lui JPEG constă în a aplica separat un **DCT (Discrete Cosine Transformation, rom: transformare cosinusoidală discretă)** pentru fiecare din cele 7200 de blocuri. Ieșirea fiecărui DCT este o matrice de 8x8 cu coeficienții DCT. Elementul DCT (0,0) este valoarea medie a blocului. Celelalte elemente spun câtă putere spectrală este prezentă la fiecare frecvență spațială. Teoretic, DCT este fără pierderi, dar în practică folosirea numerelor în virgulă mobilă și a funcțiilor transcendente introduce întotdeauna erori de rotunjire care conduc la o mică pierdere de informații. În mod normal, aceste elemente se micșorează rapid cu distanța de la origine, (0, 0), așa cum este sugerat în fig. 7-73.

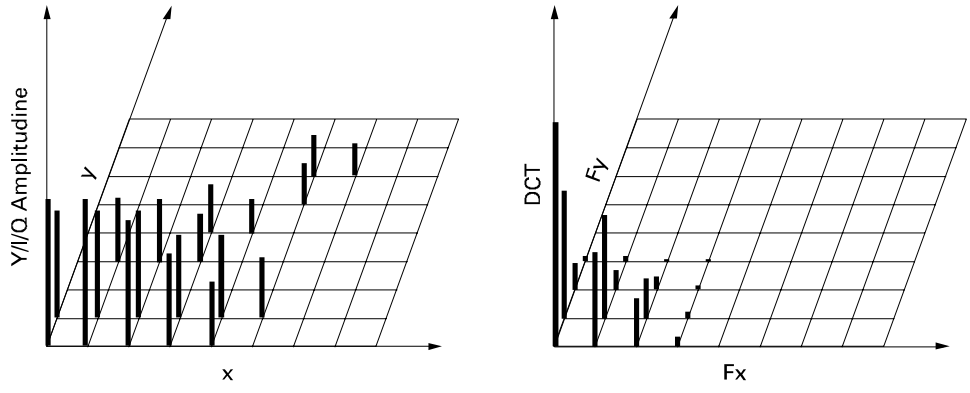


Fig. 7-73. (a) Un bloc al matricei *Y*. (b) Coeficienții DCT.

Odată ce DCT-ul este complet, JPEG trece la pasul 3, numit **cuantificare (quantization)**, în care coeficienții DCT cei mai puțin importanți sunt eliminați. Această transformare (cu pierderi) este făcută prin împărțirea fiecărui coeficient din matricea DCT de 8*8 la o pondere luată dintr-o tabelă. Dacă toate ponderile sunt 1, transformarea nu face nimic. Totuși, atunci când ponderile cresc rapid de la origine, frecvențele spațiale înalte sunt eliminate imediat.

Un exemplu al acestui pas este prezentat în fig. 7-74. Aici vedem matricea inițială DCT, tabela de cuantificare și rezultatul obținut prin împărțirea fiecărui element DCT prin elementul corespunzător din tabela de cuantificare. Valorile din tabela de cuantificare nu fac parte din standardul JPEG. Fiecare aplicație trebuie să și le furnizeze, permițând să se controleze raportul compresie/pierderi.

Coeficienți DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Coeficienți cuantificați

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabela de cuantificare

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Fig. 7-74. Calculul coeficienților DCT cuantificați.

Pasul 4 reduce valoarea (0, 0) a fiecărui bloc (cea din colțul stânga sus), înlocuind-o cu diferența față de elementul corespunzător din blocul precedent. Deoarece aceste elemente sunt mediile respectivelor blocuri, ele trebuie să se modifice lent, astfel încât considerarea valorilor diferențiale ar trebui să reducă majoritatea dintre ele la valori mici. Diferențele nu sunt calculate din celelalte valori. Valorile (0,0) sunt numite componente DC; celelalte valori sunt componente AC.

Pasul 5 liniarizează cele 64 de elemente și aplică listei codificarea după lungimea succesiunilor. Scanarea blocului de la stânga la dreapta și apoi de sus în jos nu va concentra zerourile împreună, așa încât se folosește un model de căutare în zigzag, ca în fig. 7-75. În acest exemplu, modelul în zigzag produce 38 de 0-uri consecutive la sfârșitul matricei. Acest șir poate fi redus la un singur număr spunând că sunt 38 de 0-uri, tehnică cunoscută sub numele de **codificare după lungimea succesiunilor (run-length encoding)**.

Acum avem o listă de numere care reprezintă imaginea (în spațiu transformat). Pasul 6, Huffman, codifică numerele pentru memorare sau transmitere, atribuind numerelor mai des întâlnite coduri mai scurte decât ale celorlalte numere.

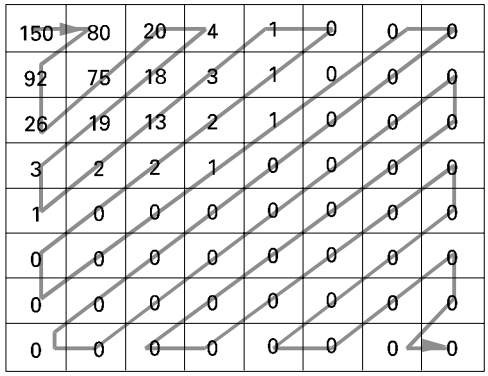


Fig. 7-75. Ordinea de transmitere a valorilor cuantificate.

JPEG poate părea complicat, aceasta pentru că el *este* cu adevărat complicat. Chiar și așa, deoarece produce o compresie de 20:1 sau mai bună, este larg folosit. Decodificarea unei imagini JPEG cere execuția algoritmului în sens invers. JPEG este aproape simetric: decodificarea ia același timp ca și codificarea. Această proprietate nu este adevărată pentru toți algoritmii de compresie, așa cum vom vedea acum.

Standardul MPEG

În fine, ajungem la miezul lucrurilor: standardele **MPEG (Motion Picture Experts Group, rom: grupul experților în filme)**. Aceștia sunt algoritmii principali folosiți pentru compresia video și sunt standarde internaționale din 1993. Deoarece filmele conțin atât imagini cât și sunete, MPEG le poate comprima pe amândouă. Am studiat deja compresia audio și a imaginilor, deci să examinăm acum compresia video.

Primul standard finalizat a fost MPEG-1 (Standard International 11172). Scopul lui a fost de a produce ieșire video de calitatea video recorder-elor (352x240 pentru NTSC) folosind o rată de biți de 1,2 Mbps. O imagine 352x240 cu 24 biți/pixel și 25 cadre/sec are nevoie de 50,7 Mbps, deci reducerea lui la 1,2 Mbps nu este în întregime trivială. MPEG-1 poate fi transmis pe linii torsadate la distanțe modeste. MPEG-1 este de asemenea folosit pentru memorarea filmelor pe CD-ROM.

Următorul standard din familia MPEG a fost MPEG-2 (Standard International 13818), care a fost proiectat inițial pentru comprimarea video de calitate de difuzare între 4 și 6 Mbps, pentru a se potrivi într-un canal de difuzare NTSC sau PAL. Mai târziu, MPEG-2 a fost extins pentru a suporta rezoluții înalte, incluzând HDTV. Este foarte cunoscut, el stând la baza DVD-ului și a televiziunii prin satelit.

Principiile de bază ale MPEG-1 și MPEG-2 sunt similare, dar detaliile sunt diferite. La o primă aproximare, MPEG-2 este un superset al lui MPEG-1, cu posibilități, formate de cadre, și opțiuni de codificare suplimentare. Vom discuta mai întâi MPEG-1 și apoi MPEG-2.

MPEG-1 are trei părți: audio, video și sistem, care le integrează pe celelalte două, ca în fig. 7-76. Codificatoarele audio și video lucrează independent, ceea ce ridică întrebarea cum se sincronizează cele două fluxuri la receptor. Această problemă se rezolvă având un ceas sistem de 90-kHz, care afișează timpul curent pentru ambele codificatoare. Aceste valori sunt pe 33 de biți, pentru a permite filmelor să ruleze 24 de ore fără depășirea valorii maxime. Aceste amprente de timp sunt incluse în ieșirea codificată și propagate spre receptor, care le poate folosi pentru sincronizare între șirurile video și audio.

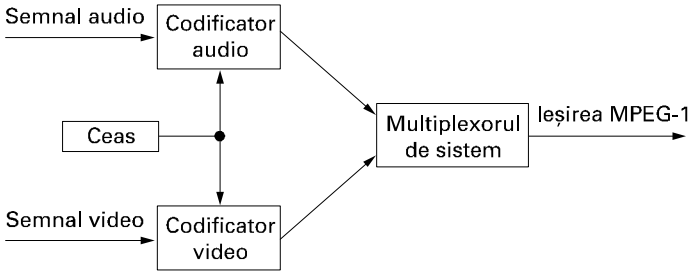


Fig. 7-76. Sincronizarea fluxurilor audio și video în MPEG-1.

Acum să considerăm compresia video MPEG-1. Există două feluri de redundanță în filme: spațială și temporală. MPEG-1 le folosește pe amândouă. Redundanța spațială poate fi folosită prin simpla codificare separată a fiecărui cadru cu JPEG. Această abordare este câteodată folosită, în special atunci când se folosesc accese aleatorii la același cadru, ca în editarea producțiilor video. În acest mod, este obținută o lărgime de bandă comprimată în intervalul 8 Mbps - 10 Mbps.

O comprimare suplimentară poate fi obținută profitând de faptul că, în general, cadrele consecutive sunt aproape identice. Acest efect este mai mic decât poate apărea la prima vedere, deoarece mulți fabricanți de filme taie scenele la fiecare 3 sau 4 secunde (cronometrați filmul și numărați scenele). Cu toate acestea, chiar o serie de 75 de cadre similare oferă potențialul unei reduceri mari față de simpla codificare separată a fiecărui cadru cu JPEG.

Pentru scene în care aparatul de filmat și fundalul sunt fixe și unul sau mai mulți actori se plimbă încet, aproape toți pixelii vor fi identici de la un cadru la altul. În acest caz, scăzând fiecare cadru din cel precedent și aplicând JPEG pe diferența lor, se obține un rezultat bun. Cu toate acestea, pentru scenele în care aparatul de filmat micșorează sau mărește, această tehnică eșuează. Este nevoie de ceva care să compenseze mișcarea. Aceasta este exact ceea ce face MPEG; este principala diferență între MPEG și JPEG.

Ieșirea MPEG-1 constă din patru tipuri de cadre:

1. Cadre I (Intracoded) : Fotografii codificate JPEG auto-conținute.
2. Cadre P (Predictive): Diferența bloc cu bloc față de ultimul cadru.
3. Cadre B (Bidirectional): Diferențele față de ultimul și de următorul cadru.
4. Cadre D (DC-coded): Medii ale blocurilor folosite pentru avans rapid.

Cadrele I sunt imagini codificate folosind JPEG, folosind de asemenea luminanța cu rezoluție completă și crominanța cu jumătate de rezoluție de-a lungul fiecărei axe. Este necesar să facem ca aceste cadre I să apară periodic în șirul de ieșire din trei motive. În primul rând, MPEG-1 poate fi folosit pentru o transmisie cu trimitere multiplă, cu vizualizatori care le acordează după dorință. Dacă toate cadrele depind de predecesoarele lor până la primul cadru, cineva care a pierdut primul cadru nu va putea decodifica cadrele succesive. În al doilea rând, dacă un cadru a fost recepționat eronat, nu este posibilă decodificarea în continuare. În al treilea rând, fără cadre I, atunci când se face un avans sau o revenire rapidă, decodificatorul ar trebui să calculeze fiecare cadru peste care trece, așa încât să știe valoarea completă a cadrului pe care este oprit. Din aceste motive, cadrele I sunt inserate la ieșire o dată sau de două ori pe secundă.

Cadrele P, în contrast, codifică diferențele între cadre. Ele se bazează pe ideea de **macroblocuri** (**macroblocks**), care acoperă 16x16 pixeli în spațiul luminanței și 8x8 pixeli în spațiul crominanței. Un macrobloc este codificat prin căutarea cadrului precedent sau ceva care diferă foarte puțin de el.

Un exemplu unde se folosesc cadrele P este redat în fig. 7-77. Aici vedem trei cadre consecutive care au același fundal, dar diferă prin poziția unei persoane. Macroblocurile care conțin scena fundalului se vor potrivi exact, dar macroblocurile conținând persoana vor fi afișate la o poziție cu o deplasare de valoare necunoscută și va trebui să fie înregistrate.

Fig. 7-77. Trei cadre consecutive.

Standardul MPEG-1 nu specifică cum trebuie făcută căutarea, cât de departe să se caute sau cât de bună trebuie să fie o potrivire pentru a conta. Aceasta depinde de fiecare implementare. De exemplu, o implementare poate căuta un macrobloc la poziția curentă din cadrul precedent și toate celelalte deplasări ale pozițiilor $\pm\Delta x$ pe direcția x și $\pm\Delta y$ pe direcția y. Pentru fiecare poziție, poate fi calculat numărul de potriviri în matricea de luminanță. Poziția cu cel mai mare scor va fi declarată câștigătoare, cu condiția să fi fost peste un prag predefinit. Altfel, macroblocul se spune că lipsește. Sunt desigur posibili și alți algoritmi mai complicați.

Dacă este găsit un macrobloc, acesta este codificat luând diferența față de valoarea sa din cadrul anterior (pentru luminanță și pentru cele două crominanțe). Aceste matrice de diferențe sunt apoi subiectul unei transformări cosinusoidale discrete, sunt cuantificate, codificate cu lungimea succesiunilor și codificate Huffman, la fel ca și cu JPEG. Valoarea pentru macrobloc în fluxul de ieșire este vectorul de mișcare (cât de departe s-a mișcat macroblocul din poziția lui precedentă pe fiecare direcție), urmată de lista de numere codificată Huffman. Dacă macroblocul nu este localizat în cadrul precedent, valoarea curentă este codificată cu JPEG, la fel ca în cadrul I.

Evident, algoritmul este puternic asimetric. O implementare este liberă să încerce fiecare poziție plauzibilă din cadrul precedent, dacă vrea s-o facă, într-o încercare disperată de a localiza fiecare ultim macrobloc, oriunde s-ar fi mișcat acesta. Această tratare va minimiza fluxul MPEG-1 codificat, cu prețul unei codificări foarte lente. Această abordare poate fi bună pentru o singură codificare a unei filmoteci, dar va fi groaznică pentru o video conferință în timp-real.

Similar, fiecare implementare este liberă să decidă ce înseamnă un macrobloc „găsit”. Această libertate permite implementatorilor să concureze prin calitatea și viteza algoritmilor proprii, dar întotdeauna produce MPEG-1. Indiferent de algoritmul de căutare folosit, ieșirea finală este ori codificarea JPEG a macroblocului curent, ori codificarea JPEG a diferenței între macroblocul curent și unul din cadrul precedent la o deplasare specificată față de cel curent.

Până acum, decodificarea MPEG-1 este directă. Decodificarea cadrelor I este la fel ca decodificarea imaginilor JPEG. Decodificarea cadrelor P cere decodificatorului să memoreze cadrul precedent și apoi să construiască noul cadru într-un nou tampon bazat pe macroblocuri codificate complet și macroblocuri care conțin diferențe față de cadrul precedent. Noul cadru este asamblat macrobloc cu macrobloc.

Cadrele B sunt similare cadrelor P, cu excepția faptului că ele permit ca macroblocul de referință să fie sau într-un cadru precedent sau în cel următor. Această libertate suplimentară permite o compensare îmbunătățită a mișcării și este de asemenea utilă atunci când obiectele trec înaintea sau în

spatele altor obiecte. Pentru a codifica cadrele B, codificatorul trebuie să țină în memorie simultan trei cadre decodificate: cel vechi, actualul și viitorul. Deși cadrele B furnizează cea mai bună compresie, nu toate implementările le suportă.

Cadrele D sunt folosite doar pentru a face posibilă afișarea imaginilor de rezoluție mică atunci când se face o revenire sau o înaintare rapidă. Realizarea decodificării MPEG-1 normală în timp-real este destul de dificilă. Cerând ca decodificatorul s-o facă atunci când se mișcă prin video de zece ori mai repede decât normal, este prea mult. În schimb, cadrele D sunt folosite pentru a produce imagini de joasă rezoluție. Fiecare intrare a cadrului D este valoarea medie a unui bloc, fără codificare ulterioară, făcând ușoară afișarea în timp-real. Această facilitate este importantă pentru a permite oamenilor să caute prin video, la viteză mare, o anumită scenă.

Terminând tratarea lui MPEG-1, să trecem la MPEG-2. Codificarea MPEG-2 este fundamental similară cu codificarea MPEG-1 cu cadre I, P, și B. Cadrele D nu sunt suportate. De asemenea, transformarea cosinusoidală discretă folosește un bloc de 10x10 în loc de 8x8, pentru a avea cu 50% mai mulți coeficienți și implicit o calitate mai bună. Deoarece este destinat televiziunii și DVD-ului, MPEG-2 suportă atât imagini progresive cât și întrețesute, în timp ce MPEG-1 suportă doar imagini progresive. Mai sunt și alte detalii minore care diferă în cele două standarde.

În loc de a suporta un singur nivel de rezoluție, MPEG-2 suportă patru: scăzut (352x240), principal (720x480), înalt-1440 (1440x1152) și înalt (1920x1080). Rezoluția scăzută este pentru VCR-uri și compatibilitate cu MPEG-1. Principalul nivel este cel normal pentru difuzarea NTSC. Celelalte două sunt pentru HDTV. Pentru o calitate foarte bună a ieșirii, MPEG-2 rulează în general la 4-8 Mbps.

7.4.8 Video la cerere

Mecanismul de video la cerere este câteodată comparat cu un magazin de închiriere a casetelor video. Utilizatorul (clientul) selectează una dintr-un număr mare de casete video pe care le are la dispoziție și o ia acasă pentru vizionare. Doar că pentru video la cerere, selecția este făcută acasă folosind telecomanda televizorului și caseta video începe imediat. Nu este necesar un drum până la magazin. Este inutil să spunem că implementarea video-ului la cerere este un pic mai complicată decât descrierea lui. În această secțiune vom face o prezentare a ideilor de bază și a implementării lor.

Este video la cerere într-adevăr precum închirierea unei casete video, sau mai degrabă precum alegerea unui film pentru vizionare la sisteme de televiziune prin cablu cu 500 de canale? Răspunsul are importante implicații tehnice. În particular, utilizatorii de casete video închiriate sunt obișnuiți cu ideea să oprească un film, să se ducă la bucătărie sau la baie și apoi să continue din locul în care caseta video a fost oprită. Telespectatorii nu se așteaptă să oprească un program.

Pentru a concura cu succes cu magazinele de închiriere, video-ul la cerere ar trebui să poată opri, porni și relua casetele video la dorință. Asigurarea acestei posibilități obligă furnizorul de video să transmită o copie separată fiecărui utilizator.

Pe de altă parte, dacă video-ul la cerere este văzut mai mult ca o televiziune avansată, atunci este suficient ca furnizorul de casete video să pornească fiecare video popular la fiecare 10 minute și să ruleze non-stop. Un utilizator care dorește să vadă un astfel de film trebuie să aștepte cel mult 10 minute pentru ca el să înceapă. Deși oprirea și repornirea nu este posibilă aici, o persoană care se întoarce în cameră după o scurtă pauză, poate comuta pe un alt canal, care prezintă aceeași casetă video, dar cu o întârziere de 10 minute. Ceva se va repeta dar nu va fi pierdut nimic. Această schemă este numită **video aproape la cerere (near video on demand)**. El oferă posibilitatea unui cost mult mai scăzut, deoarece același semnal de la furnizorul de casete video poate ajunge la mai mulți utili-

zatori odată. Diferența între video la cerere și video aproape la cerere este similară cu diferența între a circula cu mașina proprie și a lua autobuzul.

Urmărirea filmelor (aproape) la cerere este numai unul dintr-o gamă largă de noi servicii posibile de când este disponibilă rețeaua în bandă largă. Modelul general pe care mulți îl folosesc este ilustrat în fig. 7-78. În centrul sistemului se află o rețea cu coloană vertebrală de arie largă (națională sau internațională) cu lărgime de bandă mare. La ea sunt conectate mii de rețele de distribuție locală, cum ar fi cabluri TV sau sisteme distribuite ale companiei de telefoane. Sistemele de distribuție locală ajung în casele oamenilor, unde se opresc în **cutii de conectare (set-top boxes)**, care sunt, de fapt, calculatoare personale specializate.

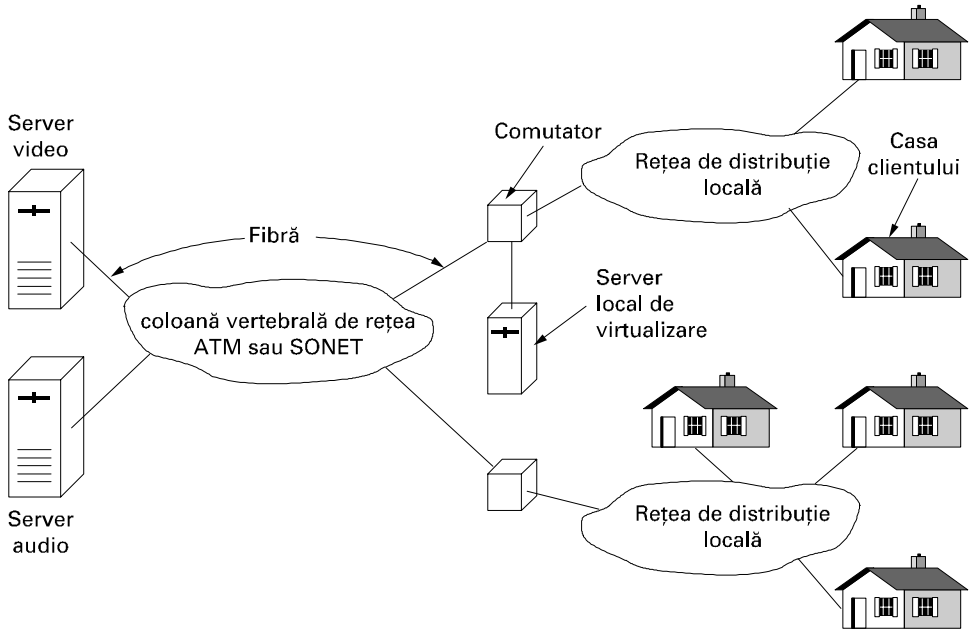


Fig. 7-78. Vedere generală a unui sistem video la cerere.

Mii de furnizori de informație sunt atașați la coloana vertebrală prin fibre optice de lărgime de bandă mare. Unii dintre aceștia vor oferi video cu plata-pe-vizualizare sau CD audio cu plata-pe-ascultare. Altele vor oferi servicii specializate, precum cumpărături de la domiciliu (cu posibilitatea de a roti o cutie de supă și a mări lista de ingrediente sau de a vedea un video-clip despre cum să conducă o mașină de cosit). Fără îndoială că în curând vor deveni disponibile sporturile, știrile, reluările la „I Love Lucy", accesul la WWW și nenumăratele alte posibilități.

În sistem sunt incluse și servere locale care permit ca video-urile să fie amplasate mai aproape de utilizatori (în avans), pentru a economisi lărgimea de bandă în timpul orelor de vârf. Cum se vor potrivi una cu alta aceste piese și cui vor aparține se va dezbate destul în industrie. În cele ce urmează, vom examina cum sunt create piesele principale din sistem: serverele video și rețeaua de distribuție.

Video-Servere

Pentru a avea video (aproape) la cerere, avem nevoie de **video-servere** (video servers) capabile să memoreze și să transmită simultan un număr mare de filme. Numărul total de filme realizate este estimat la 65.000 (Minoli, 1995). Atunci când este comprimat în MPEG-2, un film normal ocupă în

jur de 4 GB, așa că 65.000 de filme ar necesita în jur de 260 teraocteți. Adăugați la aceasta toate programele vechi de televiziune care au fost vreodată realizate, filmele de sport, jurnalele sonore, cataloagele vorbitoare pentru cumpărături etc. și este clar că ne confruntăm cu o problemă de înmagazinare foarte dificilă.

Cea mai ieftină soluție pentru memorarea unui volum mare de informație este pe bandă magnetică. Acesta a fost mereu cazul și probabil va continua să fie. O bandă Ultrium poate memora 200 GB (50 filme) la un cost de circa 1 - 2 dolari/film. Actualmente sunt comercializate servere mari, mecanice, de benzi care păstrează mii de benzi și au un braț robotic pentru extragerea fiecărei benzi și inserarea ei într-o unitate de bandă. Problema cu aceste sisteme este timpul de acces (în special pentru al 50-lea film de pe bandă), viteza de transfer și numărul limitat de unități de bandă (pentru a servi *n* filme deodată, unitatea necesită *n* unități).

Din fericire, experiența cu magazinele de închiriat video, bibliotecile publice și alte astfel de organizații arată că nu toate produsele sunt la fel de populare. Experimental, atunci când sunt disponibile *N* filme, fracția tuturor cererilor pentru filmul care ocupă locul *k* în topul popularității este de aproximativ C/k. Aici C este calculat pentru a normaliza suma la 1, cu formula:

$$C=1/(1+1/2+1/3+1/4+1/5+\ldots+1/N)$$

Astfel, filmul de pe primul loc este de șapte ori mai popular decât filmul de pe locul șapte. Acest rezultat este cunoscut drept **legea lui Zipf** (Zipf, 1949).

Faptul că unele filme sunt mai populare decât altele sugerează o soluție posibilă în forma ierarhiei de memorare, așa cum se arată în fig. 7-79. Aici, performanța crește cu cât se urcă în ierarhie.

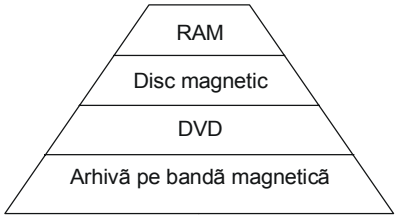


Fig. 7-79. Ierarhia video-serverului de memorare.

O alternativă la înregistrarea pe bandă este memoria optică. DVD-urile actuale memorează 4.7 GB, suficienți pentru un film, dar următoarea generație va păstra două filme. Deși timpii de căutare sunt mici în comparație cu discurile magnetice (50 ms față de 5 ms), costul lor scăzut și fiabilitatea ridicată fac din tonomatele optice care conțin mii de DVD-uri o alternativă bună față de bandă în cazul filmelor cel mai mult folosite.

Urmează discurile magnetice. Acestea au timp de acces mic (5 ms), viteza de transfer mare (320 MB/sec pentru SCSI 320), și capacități substanțiale (> 100 GB), care le fac potrivite pentru memorarea filmelor care sunt transmise efectiv (spre deosebire de simpla memorare pentru cazul că cineva ar dori vreodată să le vadă). Marele lor inconvenient este costul ridicat pentru memorarea filmelor care sunt rar accesate.

În vârful piramidei din fig. 7-79 este RAM. RAM-ul este cel mai rapid mediu de stocare, dar și cel mai scump. Atunci când prețurile la RAM ajung la 50 dolari/gigaoctet, un film de 4 GB va ocupa RAM în valoare de 200 dolari, așa că, existența a 100 de filme în RAM va costa 20.000 dolari pentru 200 GB de memorie. În plus, ideea unui video-server care transmite 100 de filme, păstrându-le pe

toate în RAM, începe să devină realizabilă. Și dacă video-serverul are 100 de clienți, dar ei se uită simultan doar la 20 de filme diferite, ideea începe să pară realizabilă, având un concept bun.

Deoarece un video-server nu este decât un imens dispozitiv de I/O în timp-real, el necesită o altă arhitectură hardware și software decât un PC sau o stație de lucru UNIX. Arhitectura hardware a unui video-server tipic este prezentată în fig. 7-80. Serverul are una sau mai multe unități centrale de înaltă performanță, fiecare cu memorie locală, o memorie principală partajată, o memorie tampon RAM de mare capacitate pentru filmele populare, o varietate de echipamente de stocare pentru păstrarea filmelor și hardware de conectare în rețea, în mod normal o interfață optică cu o rețea SONET sau ATM la viteza unui OC-12 sau mai ridicată. Aceste subsisteme sunt conectate printr-o magistrală de foarte mare viteză (cel puțin de 1 GB/sec).

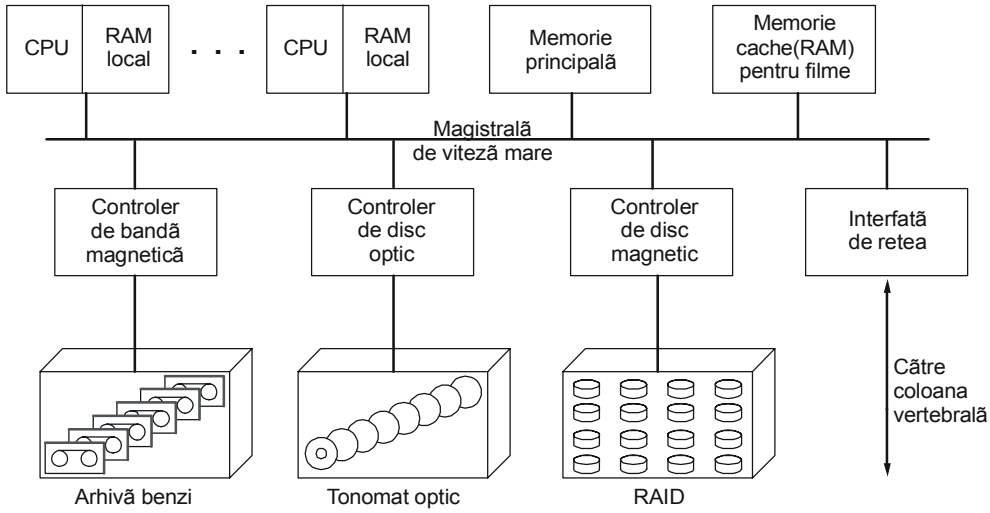


Fig. 7-80. Arhitectura hardware a unui video-server tipic.

Acum să inspectăm programele pentru video-server. Unitățile centrale sunt folosite pentru acceptarea cererilor utilizatorilor, localizarea filmelor, mutarea datelor între echipamente, taxarea clienților, și multe alte funcții. Unele dintre acestea nu sunt critice în timp, dar multe altele sunt, așa că unele, dacă nu toate unitățile centrale, va trebui să ruleze un sistem de operare de timp-real, cum ar fi un micronucleu de timp-real. În mod normal aceste sisteme împart munca în operații mai simple, fiecare cu un termen final cunoscut. Atunci planificatorul poate rula un algoritm de tip „alege termenul cel mai apropiat" sau un algoritm monoton al vitezelor (Liu și Layland, 1973).

Software-ul unității centrale definește și natura interfeței pe care serverul o prezintă clienților (servere de virtualizare și cutii de conectare). Există două tipuri populare. Primul este un sistem de fișiere tradițional, în care clienții pot deschide, citi, scrie și închide fișiere. Spre deosebire de complicațiile introduse de ierarhia de memorii și considerațiile de timp-real, un astfel de server poate avea un sistem de fișiere modelat după cel din UNIX.

Al doilea tip de interfață este bazat pe modelul video-recorderului. Comenzile adresate serverului solicită deschiderea, rularea, oprirea, derularea rapidă înainte și înapoi a fișierelor. Diferența față de modelul UNIX este că atunci când este dată o comandă PLAY, serverul continuă să transmită date la o viteză constantă, fără a necesita comenzi noi.

Inima software-ului video-serverului este sistemul de gestiune a discului. Acesta are două misiuni principale: plasarea filmelor pe discul magnetic atunci când trebuie extrase de pe bandă sau din memoria optică și tratarea cererilor către disc pentru multiplele fluxuri de ieșire. Plasarea filmelor este importantă, deoarece poate afecta foarte mult performanțele.

Două modalități de organizare a memorării pe disc sunt ferma de discuri și șirul de discuri. În cazul **fermei de discuri** (disk farm), fiecare unitate păstrează doar câteva filme întregi. Din motive de performanță și fiabilitate, fiecare film trebuie să fie prezent pe cel puțin două unități, poate chiar mai multe. Cealaltă organizare de memorare este cea de **tablou de discuri** (disk array) sau **RAID** (**Redundant Array of Inexpensive Disks** - tablou redundant de discuri ieftine), în care fiecare film este împărțit pe mai multe unități, de exemplu blocul 0 pe unitatea 0, blocul 1 pe unitatea 1, și așa mai departe, cu blocul n - 1 pe unitatea n - 1. După aceasta, ciclul se repetă, cu blocul n pe unitatea 0 și așa mai departe. Această organizare este numită **repartizare** (striping).

Un tablou de discuri repartizat are mai multe avantaje decât o fermă de discuri. Mai întâi, toate cele n unități pot rula în paralel, mărind performanța cu factorul n. În al doilea rând, poate fi făcut redundant prin adăugarea unei unități în plus la fiecare grup de n, unde unitatea redundantă conține SAU EXCLUSIV bloc-cu-bloc ale celorlalte unități, pentru a permite recuperarea completă a datelor în cazul în care o unitate se defectează. În sfârșit, problema echilibrării încărcării este rezolvată (nu este necesară plasarea manuală pentru evitarea plasării tuturor filmelor populare pe aceeași unitate). Pe de altă parte, organizarea de tip tablou de discuri este mai complicată decât ferma de discuri și mai sensibilă la defecte multiple. De asemenea este nepotrivită pentru operații ale video-recorder-ului, cum ar fi derularea rapidă a unui film înainte sau înapoi.

Cealaltă misiune a software-ului de disc este de a servi toate fluxurile de ieșire de timp-real și de a respecta constrângerile de timp ale acestora. Doar acum câțiva ani, aceasta necesita algoritmi complecși de planificare a sarcinilor, dar odată cu scăderea prețurilor la memorie, încep să devină posibile abordări mult mai simple. Pentru fiecare flux servit, este păstrată în RAM o zonă tampon (buffer) de, să zicem, 10 secunde de flux video (care înseamnă un spațiu ocupat de 5 MB). El este completat de un proces al discului și golit de un proces al rețelei. Cu 500 MB de RAM, pot fi servite 100 de fluxuri direct din RAM. Desigur, subsistemul discului trebuie să aibă o rată susținută de 50MB/sec pentru a păstra zonele tampon pline, dar un RAID construit din discuri SCSI de ultimă generație poate să îndeplinească ușor această cerință.

Rețeaua de distribuție

Rețeaua de distribuție este un set de comutatoare și linii între sursă și destinație. Așa cum vedem în fig. 7-78, ea constă dintr-o coloană vertebrală conectată la o rețea de distribuție locală. În mod obișnuit, coloana vertebrală este comutată, dar rețeaua locală nu.

Principala cerință impusă coloanei vertebrale este lărgimea de bandă mare. O altă cerință era ca fluctuația să fie scăzută, însă acum, chiar și cu cel mai mic PC este posibilă stocarea într-un tampon a 10 secunde de video de înaltă calitate MPEG-2 și prin urmare, fluctuația scăzută nu mai este o necesitate.

Distribuția locală este haotică, diferite companii încercând diferite rețele în diferite regiuni. Companiile telefonice, companiile de TV prin cablu și noii intrați sunt convinși cu toții că cel care ajunge primul va fi câștigătorul cel mare. În consecință asistăm la o proliferare a tehnologiilor instalate. În Japonia, unele companii de canalizare au intrat în afacerea Internet, susținând că ele au cele mai mari țevi în casele tuturor (introduc fibră optică prin ele, dar trebuie să fie foarte atente pe unde o scot). Cele patru scheme principale de distribuție locală pentru video la cerere sunt identificate prin acronimele ADSL, FTTC, FTTH și HFC. Le vom explica pe fiecare pe rând.

ADSL a fost primul reprezentant al industriei telefonice în loteria distribuției locale. Am studiat ADSL în cap. 2 și nu vom repeta acel material aici. Ideea este că fiecare casă din Statele Unite, Europa și Japonia are deja o pereche torsadată de cupru (pentru servicii telefonice analogice). Dacă aceste fire pot fi folosite pentru video la cerere, companiile telefonice ar putea să elimine concurența.

Problema, desigur, este că aceste fire nu pot suporta nici chiar MPEG-1 pe lungimea lor tipică de 10 km, ca să nu mai vorbim de MPEG-2. Filmele color, de înaltă rezoluție, necesită 4-8 Mbps, depinzând de calitatea dorită. ADSL nu este suficient de rapid decât pentru bucle locale scurte.

Al doilea proiect al companiei telefonice este **FTTC** (**Fiber To The Curb** - fibră către vecinătate). În FTTC, compania telefonică instalează fibră optică de la oficiul final la fiecare cartier rezidențial, terminată într-un echipament numit **ONU** (**Optical Network Unit** - unitate optică de rețea). Cele 16 bucle locale de cupru se pot termina în ONU. Aceste bucle sunt acum atât de scurte, încât este posibilă rularea duplex integrală T1 sau T2 peste ele, permițând filmele MPEG-1 și respectiv MPEG-2. În plus, deoarece FTTC este simetric, acum este posibilă video conferința pentru cei care lucrează acasă și pentru întreprinderile mici.

A treia soluție a companiei telefonice este de a introduce fibra optică în casele tuturor. Se numește **FTTH** (**Fiber To The Home** - fibră la casă). În această schemă, oricine poate avea OC-1, OC-3, sau chiar un purtător mai performant, dacă este cerut. FTTH este foarte scump, dar va deschide o gamă largă de posibilități atunci când va fi introdus. În fig. 7-63 am văzut cum oricine ar putea să aibă propriul său post de radio. Ce-ați zice de ideea ca fiecare membru al familiei să aibă propriul post de televiziune? ADSL, FTTC și FTTH sunt toate rețele locale de distribuție punct-la-punct, ceea ce nu este surprinzător dată fiind organizarea actuală a sistemului telefonic.

O abordare complet diferită este **HFC** (**Hybrid Fiber Coax** - fibră coaxială hibridă), care este soluția preferată actualmente, fiind instalată în prezent de către firmele de televiziune prin cablu. Aceasta este prezentată în Fig. 2-47(a). Povestea este următoarea. Cablurile coaxiale actuale de la 300 la 450 MHz vor fi înlocuite prin cabluri coaxiale la 750 MHz, îmbunătățind capacitatea de la 50 la 75 canale de 6 MHz la 125 canale de 6-MHz. Șaptezeci și cinci din cele 125 de canale vor fi folosite pentru transmiterea televiziunii analogice.

Cele 50 de canale noi vor fi modulate folosind QAM-256, care furnizează în jur de 40 Mbps pe canal, dând un total de 2 Gbps de lărgime de bandă nouă. Capetele vor fi mutate în cartier, așa încât fiecare cablu este doar pentru 500 de case. Simpla împărțire arată că fiecărei case îi poate fi alocat un canal dedicat de 4 Mbps, care poate fi folosit pentru un film MPEG-2.

Deși sună minunat, cere furnizorilor de cabluri să le înlocuiască pe cele existente cu cabluri coaxiale de 750 MHz, să instaleze noile capete de distribuție și să elimine toate amplificatoarele unidirecționale - pe scurt, să înlocuiască întregul sistem de TV prin cablu. În consecință, volumul de infrastructură nouă este comparabil cu ceea ce este necesar companiilor telefonice pentru FTTC. În ambele cazuri, furnizorul rețelei locale trebuie să instaleze fibra optică în cartierele rezidențiale. Din nou, în ambele cazuri, fibra se termină la un convertor optic-electric. În FTTC, segmentul final este o buclă locală punct-la-punct care folosește perechi torsadate. În HFC, segmentul final este un cablu coaxial partajat. Tehnic vorbind, aceste două sisteme nu sunt chiar atât de diferite pe cât vor să le prezinte creatorii lor.

Cu toate acestea, există o diferență reală care merită să fie amintită. HFC folosește un mediu partajat fără comutare sau dirijare. Orice informație transmisă prin cablu poate fi preluată de orice abonat fără multă zarvă. FTTC, care este complet comutat, nu are această proprietate. Ca rezultat, operatorii HFC vor ca video-serverele să trimită fluxuri criptate, așa încât clienții care nu au plătit pentru un film, să nu-l poată vedea. Operatorii FTTC nu doresc criptarea deoarece mărește complexitatea, scade per-

formanța și nu furnizează securitate suplimentară în sistemul lor. Din punctul de vedere al unei companii care rulează un video-server, este o bună idee să se cripteze sau nu? Un server folosit de o companie telefonică sau unul din subsidiarii sau partenerii săi poate să decidă să nu cripteze video-urile, pretinzând eficiența ca motiv, dar de fapt pentru a cauza pierderi economice competitorilor HFC.

Pentru toate rețelele locale de distribuție, este posibil că fiecare cartier va fi echipat cu unul sau mai multe servere de virtualizare. Acestea sunt, de fapt, doar versiuni mai mici ale video-serverelor despre care am discutat înainte. Marele avantaj al acestor servere locale este că reduc încărcarea coloanei vertebrale.

Ele pot fi preîncărcate cu filme fie dinamic, fie prin rezervare. Dacă oamenii spun furnizorului în avans ce filme doresc, ele pot fi transferate pe serverul local în afara orelor de vârf. Această observație orientează operatorii de rețea spre atragerea personalului de la companiile aeriene pentru stabilirea tarifelor. Se pot imagina tarife în care filmele cerute cu 24 până la 72 de ore în avans pentru a fi vizionate marțea sau joia înainte de 6 seara sau după 11 seara, primesc o reducere de 27 la sută. Filmele comandate în prima duminică a lunii înainte de 8 dimineața pentru a fi vizionate miercuri după-amiaza, într-o zi a cărei dată este un număr prim, beneficiază de o reducere de 43 la sută și așa mai departe.

7.4.9 MBone - Coloana vertebrală pentru trimitere multiplă

În timp ce toate aceste industrii fac planuri mari - și îndelung mediatizate - pentru viitorul video la cerere (inter)național, digital, comunitatea Internet și-a implementat propriul sistem multimedia digital, **MBone** (**Multicast Backbone -** coloana vertebrală cu trimitere multiplă). În această secțiune vom face o scurtă sinteză a ceea ce este și cum funcționează.

MBone poate fi gândit ca radio și televiziune Internet. Spre deosebire de video la cerere, unde accentul cade pe selectarea și vizualizarea filmelor precomprimate memorate pe un server, MBone este folosit pentru difuzare audio și video în formă digitală în lumea întreagă prin Internet. Este operațional de la începutul lui 1992. Multe conferințe științifice, în special întâlniri IETF, au fost difuzate, la fel ca și evenimentele științifice notabile, cum ar fi lansarea navetelor spațiale. Prin MBone a fost difuzat un concert Rolling Stones, precum și porțiuni din Festivalul de film de la Cannes. Este discutabil dacă acesta poate fi calificat drept un eveniment științific.

Din punct de vedere tehnic, MBone este o rețea virtuală situată deasupra Internet-ului. Ea constă din insule cu posibilități de trimitere multiplă, conectate prin tuneluri, așa cum se arată în fig. 7-81. În această figură, MBone constă din șase insule, de la *A* la *F*, conectate prin șapte tuneluri. Fiecare insulă (de obicei un LAN sau un grup de LAN-uri interconectate) suportă trimitere multiplă hardware către calculatoarele gazdă. Tunelurile propagă pachetele MBone între insule. Cândva, în viitor, când toate ruterele vor fi capabile să gestioneze direct traficul cu trimitere multiplă, această superstructură nu va mai fi necesară, dar pentru moment este funcțională.

Fiecare insulă conține unul sau mai multe rutere specializate numite **m-rutere** (**mrouters** - rutere cu trimitere multiplă). Câteva dintre acestea sunt rutere normale, dar majoritatea sunt numai stații UNIX care rulează software-ul special de nivel utilizator (dar ca supervizor). M-ruterele sunt conectate logic prin tuneluri. Pachetele MBone sunt încapsulate în pachete IP și trimise ca pachete obișnuite cu trimitere unică la adresa IP a m-ruterului destinație.

Tunelurile sunt configurate manual. În mod uzual, un tunel este o cale pentru care există o conexiune fizică, dar aceasta nu este o cerință. Dacă, accidental, calea fizică asociată unui tunel se defectează, m-ruterele care folosesc tunelul nu vor observa, deoarece Internet-ul va redirija automat întregul trafic IP dintre ele prin alte linii.

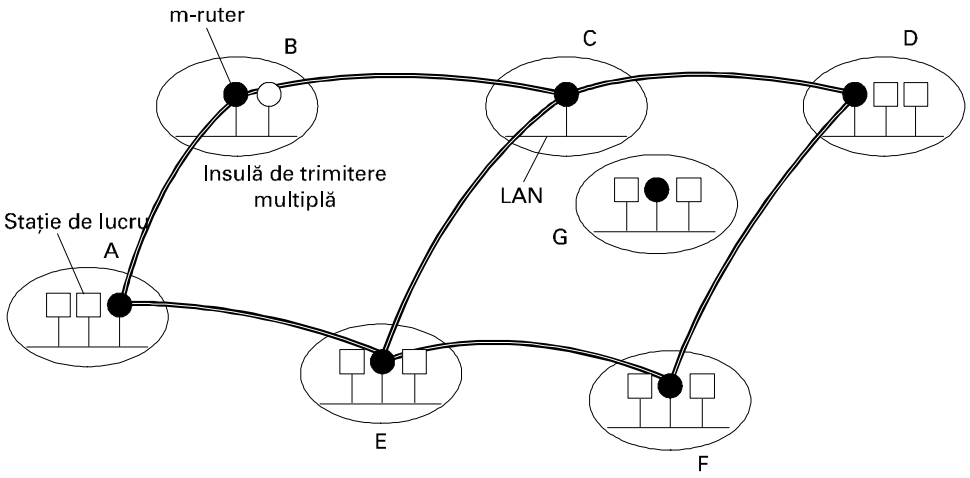


Fig. 7-81. MBone constă din insule de trimitere multiplă conectate prin tunele.

Atunci când apare o nouă insulă și dorește să se atașeze la MBone, precum *G* din fig. 7-81, administratorul său trimite un mesaj anunțând existența acesteia către lista de poștă a MBone-ului. Administratorii siturilor apropiate îl contactează pentru a stabili tunelurile. Câteodată tunelurile existente sunt reconfigurate, astfel încât să profite de noua insulă pentru a optimiza topologia. La urma urmelor, tunelurile nu au existență fizică. Ele sunt definite prin tabele în m-rutere și pot fi adăugate, șterse, sau mutate prin simpla schimbare a acestor tabele. Tipic, fiecare țară din MBone are o coloană vertebrală cu insule regionale atașate acesteia. În mod normal, MBone este configurat cu unul sau două tuneluri care traversează oceanele Atlantic și Pacific, aducând MBone-ul la scară globală.

Astfel, în orice moment, MBone constă dintr-o topologie specifică alcătuită din insule și tuneluri, independent de numărul adreselor de trimitere multiplă utilizate curent și de cine le ascultă sau le urmărește. Această situație este foarte asemănătoare cu cea a unei subrețele normale (fizice), așa încât i se aplică algoritmii normali de dirijare. În consecință, MBone a folosit inițial un algoritm de dirijare, **DVMRP (Distance Vector Multicast Routing Protocol -** dirijare multi-destinație după vectorul distanțelor) bazat pe algoritmul vectorului distanțelor al lui Bellman-Ford. De exemplu, în fig. 7-81, insula *C* poate dirija către *A* prin *B* sau prin *E* (sau prin *D*). Își alege varianta luând valorile pe care i le dau nodurile despre distanțele de la ele până la *A* și apoi adăugând distanța proprie până la ele. În acest mod, fiecare insulă determină ruta optimă către fiecare altă insulă. Totuși, rutele nu sunt de fapt folosite în acest mod, ci așa cum vom vedea în curând.

Să considerăm acum modul în care se realizează trimiterea multiplă. Pentru a transmite la mai multe destinații un program audio sau video, o sursă trebuie să achiziționeze mai întâi o adresă de destinație multiplă de clasă D, care acționează ca o frecvență de stație sau un număr de canal. Adresele de clasă D sunt rezervate prin folosirea unui program care caută într-o bază de date adrese de destinație multiplă libere. Multe trimiteri multiple pot să se desfășoare în același timp, iar un calculator gazdă poate „să se acordeze" pe cea de care este interesat prin ascultarea adrese de destinație multiplă potrivite.

Periodic, fiecare m-ruter trimite un pachet de difuzare IGMP limitat la insula lui, întrebând cine este interesat de ce canal. Calculatoarele gazdă care doresc să (continue să) primească unul sau mai

multe canale trimit alt pachet IGMP înapoi drept răspuns. Aceste răspunsuri sunt dispersate în timp pentru a evita supraîncărcarea LAN-ului local. Fiecare m-ruter păstrează o tabelă cu canalele care trebuie puse în LAN-ul propriu, pentru a evita pierderea de lărgime de bandă prin canale de transmitere multiplă pe care nu le vrea nimeni.

Trimiterile multiple se propagă prin MBone după cum urmează. Când o sursă audio sau video generează un nou pachet, îl difuzează către insula sa locală folosind facilitățile hardware de trimitere multiplă. Acest pachet este preluat de un m-ruter local, care îl copiază pe toate tunelurile cu care este conectat.

Fiecare m-ruter care primește un astfel de pachet printr-un tunel verifică dacă pachetul a venit pe cea mai bună rută, adică cea pe care tabela proprie îi spune să o folosească pentru a ajunge la sursă (ca și cum ar fi o destinație). Dacă pachetul a venit pe ruta cea mai bună, m-ruterul îl copiază la toate celelalte tuneluri. Dacă pachetul a ajuns pe o cale care nu este cea optimală, el este eliminat. Astfel, de exemplu, în fig. 7-81, dacă tabela lui *C* spune să se folosească *B* pentru a ajunge la *A*, atunci când un pachet cu trimitere multiplă de la *A* ajunge la *C* prin *B,* el este copiat la *D* și *E*. Cu toate acestea, atunci când un pachet cu trimitere multiplă de la *A* ajunge la *C* prin *E* (nu este calea cea bună), el este eliminat. Acest algoritm este algoritmul retransmiterii pe calea inversă prezentat în Cap. 5. Deși nu este perfect, este destul de bun și foarte ușor de implementat.

În plus față de folosirea algoritmului de căutare pe calea inversă, pentru prevenirea inundării Internet-ului și pentru a limita domeniul trimiterii multiple, este folosit câmpul IP *Time_to_live* (timp de viață). Fiecare pachet pleacă cu o anumită valoare (determinată de sursă). Fiecărui tunel i se asociază o pondere. Un pachet este trecut printr-un tunel dacă are o pondere suficientă. Altfel este eliminat. De exemplu, tunelurile transoceanice sunt configurate în mod obișnuit cu o pondere de 128, așa încât pachetele pot fi limitate la continentul de origine dându-li-se un timp de viață inițial mai mic sau egal cu 127. După trecerea printr-un tunel, câmpul *Time to live* este decrementat cu ponderea tunelului.

De când funcționează algoritmul de dirijare MBone, s-au făcut multe cercetări pentru a-l îmbunătăți. O propunere păstrează ideea dirijării după vectorul distanțelor, dar face algoritmul ierarhic, prin gruparea siturilor MBone în regiuni și dirijarea în prima etapă către acestea (Thyagarajan și Deering, 1995).

O altă propunere este de a folosi o formă modificată a dirijării în funcție de starea legăturilor, în loc de dirijare după vectorul distanțelor. În particular, un grup de lucru IETF se ocupă de modificarea OSPF pentru a-l face potrivit pentru trimitere multiplă în cadrul unui singur sistem autonom. Trimiterea multiplă OSPF rezultată este numită **MOSPF** (Moy, 1994). Modificările se referă la crearea de către MOSPF a unei hărți complete care, în plus față de informația vizuală de dirijare, să țină evidența insulelor de trimitere multiplă și a tunelurilor. Înarmată cu această topologie completă, este ușor să calculăm cea mai bună cale de la fiecare insulă către fiecare altă insulă folosind tunelurile. De exemplu, poate fi folosit algoritmul lui Dijkstra.

A doua arie de cercetare este dirijarea inter-AS. Aici un alt grup de lucru IETF dezvoltă un algoritm numit **PIM (Protocol Independent Multicast -** transmitere multiplă independentă de protocol). PIM are două versiuni, după cum insulele sunt dense (aproape oricine vrea să se uite) sau rare (aproape nimeni nu vrea să se uite). Ambele versiuni folosesc tabele de dirijare standard cu trimitere unică în loc de a crea o topologie suprapusă, așa cum fac DVMRP sau MOSPF.

În PIM-DM (modul dens), ideea este de a tăia căile inutile. Tăierea funcționează în modul următor. Atunci când un pachet cu trimitere multiplă ajunge printr-un tunel „greșit", un pachet de tăiere este trimis înapoi prin tunel, spunând emițătorului să nu-i mai transmită pachete de la sursa

respectivă. Când un pachet ajunge prin tunelul „bun”, este copiat pe toate celelalte tuneluri care nu s-au auto-tăiat anterior. Dacă toate celelalte tuneluri s-au auto-tăiat și canalul din insula locală nu este interesat, m-ruterul trimite un mesaj de tăiere înapoi prin canalul „bun”. În acest fel, trimiterea multiplă se adaptează automat și merge doar unde este dorită.

PIM-SM (modul rar), descris în RFC 2362, lucrează diferit. Aici ideea este de a preveni saturarea Internetului, doar pentru că trei persoane din Berkeley vor să țină o conferință peste o adresă de clasă D. PIM-SM funcționează prin fixarea unor puncte de întâlnire. Fiecare dintre sursele dintr-un grup cu trimitere multiplă PIM-SM își trimite pachetele la punctele de întâlnire. Orice sit interesat în atașare, cere unui punct de întâlnire să-i seteze un tunel. În acest mod, tot traficul PIM-SM este transportat prin transmitere simplă, în loc de transmitere multiplă. PIM-SM devine tot mai popular și MBONE migrează către folosirea lui. Pe măsură ce PIM-SM devine mai mult folosit, MOSPF dispare treptat. Pe de altă parte, însuși MBONE pare într-un fel să stagneze și probabil niciodată nu va deveni foarte popular.

Totuși, multimedia prin rețea este încă un domeniu foarte interesant, care evoluează rapid, chiar dacă MBONE nu devine un succes uriaș. Zilnic sunt anunțate noi tehnologii și aplicații. Mai mult, transmiterea multiplă și calitatea serviciului funcționează împreună, după cum se prezintă în (Striegel și Manimaran, 2002). Alt subiect fierbinte este transmisia multiplă fără fir (wireless) (Gossain et. al., 2002). Întregul domeniu al transmisiunilor multiple și orice este legat de el va rămâne, probabil, important pentru următorii ani.

7.5 REZUMAT

Atribuirea numelor în Internet folosește o schemă ierarhică, numită sistemul numelor de domenii (DNS). La nivelul superior, există bine cunoscutele domenii generice, incluzând *com* și *edu*, precum și cele aproximativ 200 domenii pentru țări. DNS este implementat ca un sistem de baze de date distribuite, cu servere în întreaga lume. DNS păstrează înregistrări cu adrese IP, centre de mesagerie și alte informații. Prin interogarea unui server DNS, un proces poate stabili corespondența dintre un nume de domeniu Internet și o adresă IP folosită pentru a comunica cu acel domeniu.

E-mail este una din cele două aplicații foarte populare din Internet. Oricine, de la copii la bunici, o poate folosi acum. Cele mai multe sisteme de poștă electronică din lume folosesc sistemul definit în RFC 2821 și 2822. Mesajele trimise în acest sistem folosesc antete de sistem ASCII pentru definirea proprietăților mesajului. Materiale cu diverse tipuri conținut pot fi transmise folosind MIME. Mesajele sunt transmise folosind SMTP, care lucrează făcând o conexiune TCP de la sistemul sursă la cel de destinație și livrând în mod direct e-mail-ul peste conexiunea TCP.

O altă aplicație foarte populară pentru Internet este World Wide Web. Web-ul este un sistem pentru legarea documentelor "hipertext". Inițial, fiecare document era o pagină scrisă în HTML, cu posibile hiper-legături la alte documente. Azi, XML câștigă teren în fața HTML. De asemenea, un mare volum de informație este generat dinamic, folosind script-uri executate de server (eng.:server-side scripts) (PHP, JSP și ASP), precum și script-uri executate de client (eng.: client-side scripts) (de remarcat aici Javascript). Un program de navigare poate afișa documentul stabilind o conexiune TCP cu serverul său, cerând documentul și apoi închizând conexiunea. Aceste mesaje de cerere conțin o mulțime de antete pentru asigurarea informației suplimentare. Folosirea memoriei ascunse,

replicarea și rețelele de livrare a conținutului sunt folosite pe scară largă pentru a îmbunătăți performanțele Web-ului.

Web-ul fără fir este de-abia la început. Primele sisteme sunt WAP și i-mode, fiecare cu ecrane mici și lungime de bandă limitate, dar cele din generația următoare vor fi mai puternice.

Multimedia este și ea o stea pe firmamentul rețelelor. Se permite ca semnalele video și audio să fie digitizate și transportate electronic pentru afișare. Audio necesită mai puțină lărgime de bandă, astfel că este mai avansat. Fluxurile audio, radio prin Internet și vocea prin IP (voice over IP) sunt acum o realitate, iar noile aplicații apar permanent. Video la cerere este un domeniu de viitor, de mare interes. În sfârșit, Mbone este un serviciu experimental, bazat pe televiziunea și radioul digital trimise peste Internet.

7.6 PROBLEME

1. Multe calculatoare ale unor firme au trei identificatori universali, unici. Care sunt ei?

2. După informațiile date în fig. 7-3, *little-sister.cs.vu.nl* se află într-o rețea de clasă A, B, sau C?

3. În fig. 7-3, nu este nici un punct după *rowboat*? De ce nu?

4. Ghiciți ce ar putea să însemne smiley-ul :-X (uneori scris ca :-#).

5. DNS folosește UDP în loc de TCP. Pachetele DNS pierdute nu pot fi recuperate automat. Cauzează acest lucru probleme, și dacă da, cum sunt ele rezolvate?

6. În plus față de problema pierderilor, pachetele UDP au o dimensiune maximă, uneori ajungând chiar la minimum 576 octeți. Ce se întâmplă când numele DNS căutat depășește această dimensiune? Poate fi trimis în două pachete?

7. Se poate ca o mașină cu un singur nume DNS să aibă mai multe adrese IP? Cum ar putea să se întâmple acest lucru?

8. Este posibil ca un calculator să aibă două nume DNS care aparțin de două domenii de nivel înalt? Dacă da, dați un exemplu plauzibil. Daca nu, explicați de ce.

9. Numărul de companii cu site Web a crescut exploziv în ultimii ani. Ca rezultat, mii de companii au fost înregistrate în domeniul *com*, ducând la o încărcare mare a serverului pentru acest domeniu. Sugerați o cale de a diminua această problemă fără a schimba schema de nume (adică fără a introduce noi domenii de nivel înalt). Este permis ca soluția să impună schimbarea codului de la client.

10. Unele sisteme de e-mail suportă în antet câmpul *Content Return:*. El specifică dacă corpul mesajului trebuie să fie returnat în cazul imposibilității livrării. Acest câmp aparține plicului sau conținutului?

11. Sistemele de poștă electronică au nevoie de registre pentru a putea căuta adresele de e-mail. Pentru a construi asemenea registre, numele ar trebui să fie separate în componentele standard (de exemplu nume, prenume) pentru a putea fi făcute căutări. Discutați unele dintre problemele ce trebuie rezolvate pentru acceptarea universală a unui astfel de standard.

12. Adresa de e-mail a unei persoane este numele său de utilizator @ numele DNS cu o înregistrare MX. Numele de utilizator pot fi prenume, nume, inițiale, tot felul de alte nume. Să presupunem că o firmă mare a decis că se pierdea prea mult e-mail din cauză că lumea nu cunoștea numele de utilizator al destinatarului. Există vreo modalitate pentru ca ei să rezolve această problemă fără schimbarea DNS-ului? Dacă da, dați o propunere și explicați cum funcționează. Dacă nu, explicați de ce este imposibil.

13. Un fișier binar are lungimea de 3072 de biți. Cât de lung va fi dacă îl codificăm folosind base64, cu perechea CR+LF inserată după fiecare 80 de octeți transmiși și la sfârșit?

14. Considerați schema de codificare MIME afișabilă-marcată. Menționați o problemă nediscutată în text și propuneți o soluție.

15. Dați cinci tipuri MIME care nu sunt listate în text. Puteți să verificați browser-ul dvs. sau să căutați pe Internet.

16. Să presupunem că vreți să trimiteți un fișier MP3 la un prieten, dar ISP-ul prietenului limitează dimensiunea unui mesaj la 1 MB iar fișierul MP3 are 4 MB. Există vreo modalitate de a rezolva problema aceasta folosind RFC 822 și MIME?

17. Să presupunem că cineva instalează un demon de vacanță (vacation daemon) și apoi trimite un mesaj chiar înainte de a ieși din sistem. Din păcate, destinatarul este în vacanță de o săptămână și are de asemenea instalat un demon de vacanță. Ce se întâmplă în continuare? Replicile vor fi trimise dintr-o parte în alta până când se va întoarce cineva?

18. În orice standard, ca de exemplu RFC 822, este necesară o gramatică precisă a ceea ce este permis astfel încât implementări diferite să poată conlucreze. Chiar și unitățile simple trebuie să fie definite cu atenție. Antetele SMTP permit existența spațiului între simboluri. Dați *două* definiri plauzibile ale spațiului dintre simboluri.

19. Demonul de vacanță este parte a agentului utilizator sau a agentului de transfer? Desigur, este instalat folosind agentul utilizator, dar cel care trimite replicile este chiar agentul utilizator? Explicați răspunsul.

20. POP3 permite utilizatorilor să aducă mesajele de e-mail dintr-o cutie poștală de la distanță. Aceasta înseamnă că formatul intern al cutiilor poștale trebuie să fie standard pentru ca orice program POP3 de la client să poată să citească cutia poștală de pe orice server de poștă electronică? Discutați răspunsul.

21. Din punctul de vedere al unui ISP, POP3 și IMAP diferă într-o măsură importantă. Utilizatorii POP3 își golesc în general cutiile poștale zilnic. Utilizatorii IMAP își păstrează mesajele pe server un timp nedefinit. Imaginați-vă că vi se cere să sfătuiți un ISP ce protocol ar trebui să suporte. Ce argumente ați aduce?

22. Poșta pe Web(Webmail) folosește POP3, IMAP sau nici unul? Dacă folosește unul din ele, de ce a fost ales acela? Dacă nici unul, care este mai aproape de idee?

23. Când sunt transmise, paginile de Web sunt prefixate de antete MIME. De ce?

24. Când sunt necesare programe de vizualizare externe? Cum știe un program de navigare pe care să-l folosească?

25. Este posibil ca atunci când un utilizator urmează pe o hiper-legătură în Netscape, să fie pornit un anumit program, iar urmând aceeași hiper-legătură în Internet Explorer să fie pornit un program complet diferit, chiar dacă tipul MIME întors în ambele cazuri este identic? Explicați răspunsul.

26. Un server Web cu mai multe fire de execuție este organizat ca în fig. 7-21. Durează 500 μsec să accepte o cerere și să verifice memoria ascunsă. Jumătate din timp, fișierul este găsit în memoria ascunsă și este întors imediat. Cealaltă jumătate, modulul trebuie să se blocheze 9 ms până când cererea la disc este adăugată în coadă și procesată. Câte module ar trebui să aibă serverul pentru a ține procesorul ocupat tot timpul (presupunând că discul nu reprezintă o gâtuire (bottleneck))?

27. URL-ul standard *http* presupune că serverul de Web ascultă pe portul 80. Totuși, e posibil ca un server de Web să asculte pe alt port. Născociți o sintaxă rezonabilă pentru URL pentru accesarea unui fișier pe un port nestandard.

28. Cu toate că nu a fost menționată în text, o formă alternativă pentru un URL este folosirea adresei IP în loc de numele său DNS. Un exemplu de folosirea a adresei IP este *http://192.31.231.66/index.html*. Cum știe programul navigator dacă numele ce urmează schema este un nume DNS sau o adresă IP?

29. Imaginați-vă că cineva de la Departamentul CS din Stanford a scris un nou program pe care vrea să-l distribuie prin FTP. El pune programul în catalogul */ftp/pub/freebies/newprog.c*. Care este URL-ul probabil pentru acest program?

30. În fig. 7-25, *www.aportal.com* ține evidența preferințelor utilizatorilor într-un cookie. Un dezavantaj al acestei scheme este că cookie-urile sunt limitate la 4KB și dacă preferințele sunt extinse, de exemplu la multe valori ale acțiunilor, echipe sportive, tipuri de știri, vremea în multe orașe, ofertele din diverse categorii de produse și altele, limita de 4KB ar putea fi insuficientă. Proiectați o alternativă pentru păstrarea preferințelor utilizatorului pentru a nu avea această problemă.

31. Banca Sloth (Trândăvie) dorește să facă operațiile bancare mai simple pentru utilizatorii mai leneși, astfel încât după ce un utilizator se autentifică, banca îi returnează identificatorul de client într-un cookie. Ce părere aveți de această idee? Va funcționa? Este o idee buna?

32. În fig. 7-26, în marcajul <IMG> apare parametrul *ALT*. În ce condiții este folosit de programul de navigare și cum?

33. Realizați o imagine selectabilă în HTML? Dați un exemplu.

34. Arătați cum marcajul <a> poate fi folosit pentru a face șirul „ACM” un hiper-legături către *http://www.acm.org*.

35. Proiectați un formular pentru o nouă companie, InterBurger, care permite comanda hamburgerilor prin Internet. Formularul trebuie să conțină numele clientului, adresa, orașul, ca și o opțiune asupra dimensiunii (ori gigant, ori imens) și o opțiune pentru brânză. Burger-ii urmează a fi plătiți la livrare cu bani gheață, așa că nu este necesară nici o informație despre cartea de credit.

36. Proiectați un formular care cere utilizatorului să tasteze două numere. Când utilizatorul apasă pe butonul de trimitere, serverul întoarce suma lor. Scrieți partea care rulează la server ca un script PHP.

37. Pentru fiecare din aplicațiile următoare, spuneți (1) dacă este posibil și (2) dacă este mai bine să se folosească un script PHP sau JavaScript și de ce.
(a) Afișarea unui calendar pentru orice lună începând cu septembrie 1752.
(b) Afișarea unui program al zborurilor de la Amsterdam la New York.
(c) Graficul unui polinom cu coeficienții dați de utilizator.

38. Scrieți un program JavaScript care acceptă un întreg mai mare ca 2 și spune dacă este, sau nu, un număr prim. Notați că JavaScript are instrucțiunile if și while cu aceeași sintaxă ca în C sau Java. Operatorul modul este %. Dacă aveți nevoie de rădăcina pătrată a lui *x*, folosiți *Math.sqrt(x)*.

39. O pagină HTML este de forma:

```
<html> <body>
<a href="www.info-source.com/welcome.html"> Apăsați aici pentru informații </a>
</body> </html>
```

Dacă utilizatorul apasă pe hiper-legătură, este deschisă o conexiune TCP și este trimisă o serie de linii la server. Scrieți toate liniile trimise.

40. Antetul *If-Modified-Since* poate fi folosit pentru a vedea dacă o pagină din memoria ascunsă este încă validă. Cererile pot fi pentru pagini conținând imagini, sunete, video etc., precum și HTML. Credeți că eficacitatea acestei tehnici este mai bună sau mai rea pentru imagini JPEG în comparație cu HTML? Gândiți bine la ceea ce „eficacitate” înseamnă și explicați răspunsul vostru.

41. În ziua unui mare eveniment sportiv, cum ar fi un campionat sportiv, multă lume se duce pe situl Web oficial. Este aceasta o aglomerare bruscă în același sens cu cel al alegerilor din Florida, în 2000? De ce sau de ce nu?

42. Are sens ca un singur ISP să aibă rolul de CDN? Dacă da, cum ar funcționa? Dacă nu, care este greșit în legătură cu această idee?

43. În ce condiții folosirea unui CDN este o idee proastă?

44. Terminalele pentru Web-ul fără fir(Wirelss Web) au o lățime de bandă mică, ceea ce face importantă o codificare eficientă. Proiectați o schemă de transmitere eficientă a textului în engleză pe o legătură fără fir către un dispozitiv WAP. Puteți presupune că terminalul are câțiva mega-

octeți de memorie ROM și un procesor moderat de puternic. *Indiciu*: gândiți-vă cum transmiteți ceva în japoneză, unde fiecare simbol este un cuvânt.

45. Un CD memorează 650 MB de date. Este folosită compresia pentru CD-uri audio? Explicați raționamentul.

46. În fig. 7-57(c) zgomotul de cuantificare apare datorită folosirii de eșantioane pe 4 biți pentru a reprezenta nouă valori de semnale. Primul eșantion, la 0, este exact, dar câteva dintre următoarele nu. Care este procentul de eroare la 1/32, 2/32 și 3/32 din perioadă?

47. Ar putea fi folosit modelul psiho-acustic pentru reducerea lărgimii de bandă necesare pentru telefonia Internet? Dacă da, ce condiții, dacă există, ar trebui să fie îndeplinite pentru ca el să funcționeze? Dacă nu, de ce nu?

48. Un server de flux audio are o distanță pe sens de 50 ms cu un dispozitiv de redare (media plsyer). El emite la 1 Mbps. Dacă media player-ul are o memorie tampon de 1 MB, ce puteți spune despre poziția minimă și cea maximă?

49. Algoritmul de întrețesere din fig. 7-60 are avantajul de a fi capabil să supraviețuiască unei pierderi ocazionale a unui pachet fără a introduce o pauză în redarea sunetului (playback). Totuși, când este folosit pentru telefonia Internet, el are și un mic dezavantaj. Care este el?

50. Transmisia de voce-peste-IP are aceleași probleme cu zidurile de protecție (firewalls) ca și transmiterea fluxurilor audio? Discutați răspunsul vostru.

51. Care este rata de biți pentru transmiterea necomprimată a culorii la 800 x 600 cu 8 biți/pixel la 40 cadre/sec?

52. Poate o eroare de 1-bit într-un cadru MPEG să afecteze mai mult decât cadrul în care a apărut eroarea? Explicați răspunsul vostru.

53. Să considerăm exemplul video-serverului cu 100.000 de clienți, unde fiecare client vizionează două filme pe lună. Jumătate din filme se transmit la 8 seara. Câte filme trebuie să transmită serverul simultan în acest interval de timp? Dacă fiecare film necesită 4 Mbps, câte conexiuni OC-12 necesită serverul pentru rețea?

54. Să presupunem că legea lui Zipf este îndeplinită pentru accese la un server video cu 10.000 de filme. Dacă serverul memorează cele mai populare 1000 de filme pe disc magnetic, iar restul de 9000 pe disc optic, dați o expresie pentru fracția tuturor referințelor care se vor face la discul magnetic. Scrieți un mic program pentru evaluarea acestei expresii numerice.

55. Unii cyber-băgăcioși și-au înregistrat nume de domenii care sunt ortografieri greșite ale siturilor companiilor cunoscute, ca de exemplu, *www.microsfot.com*. Faceți o listă de cel puțin cinci asemenea domenii.

56. Numeroase persoane au înregistrat nume de domenii DNS de genul *www.cuvânt.com*, unde *cuvânt* este un cuvânt obișnuit. Pentru fiecare din categoriile următore, enumerați cinci situri Web și spuneți pe scurt despre ce este vorba (de exemplu, *www.stomach.com* este un gastroenteorologist din LongIsland). Iată lista de categorii: animale, mâncare, obiecte de gos-

podărie, părți ale corpului. Pentru ultima categorie, vă rog rămâneți la părțile corpului de deasupra taliei.

57. Proiectați emoji-uri proprii folosind o hartă de biți 12 x 12. Includeți prietenul, prietena, profesorul și politicianul.

58. Scrieți un server POP3 care acceptă următoarele comenzi: *USER, PASS, LIST, RETR, DELE* și *QUIT*.

59. Rescrieți serverul din fig. 6-6 ca un server Web adevărat, folosind comanda GET de la HTTP 1.1. Ar trebui să accepte și mesajul *Host*. Serverul trebuie să mențină o memorie ascunsă cu fișierele recent aduse de pe disc și să servească cererile din această memorie atunci când este posibil.

8

SECURITATEA REȚELELOR

În primele decenii ale existenței lor, rețelele de calculatoare au fost folosite de cercetătorii din universități pentru trimiterea poștei electronice și de către funcționarii corporațiilor pentru a partaja imprimantele. În aceste condiții, problema securității nu atrăgea prea mult atenția. Dar acum, când milioane de cetățeni obișnuiți folosesc rețelele pentru operațiuni bancare, cumpărături și plata taxelor, securitatea rețelei apare la orizont ca o mare problemă potențială. În acest capitol, vom studia securitatea rețelei din mai multe unghiuri, evidențiind numeroase pericole și discutând mulți algoritmi și protocoale destinate a face rețelele mai sigure.

Securitatea este un subiect vast și acoperă o multitudine de imperfecțiuni. În forma sa cea mai simplă, ea asigură că persoane curioase nu pot citi sau, și mai rău, modifica mesajele adresate altor destinatari. Ea se ocupă de cei care încearcă să apeleze servicii la distanță, deși nu sunt autorizați să le folosească. De asemenea, securitatea implică verificarea dacă un mesaj, ce pretinde că vine de la IRS și spune: „Plătește până vineri”, provine într-adevăr de la IRS și nu de la Mafie. Securitatea se ocupă de problemele legate de capturarea și falsificarea mesajelor autorizate și de cei ce încearcă să nege faptul că au trimis anumite mesaje.

Majoritatea problemelor de securitate sunt cauzate intenționat de persoane răuvoitoare care încearcă să obțină anumite beneficii, să atragă atenția, sau să provoace rău cuiva. Câțiva dintre cei care comit în mod obișnuit astfel de fapte sunt menționați în fig. 8-1. Din această listă trebuie să rezulte clar că realizarea unei rețele sigure implică ceva mai mult decât păstrarea ei fără erori de programare. Aceasta implică surclasarea unor adversari adeseori inteligenți, dedicați și uneori bine dotați material. Trebuie de asemenea să fie clar că măsurile care pot contracara inamici accidentali vor avea un impact redus asupra unor adversari serioși. Arhivele poliției arată că cele mai multe atacuri nu au fost săvârșite de străini prin ascultarea unor linii telefonice, ci de către angajați ranchiunoși. În consecință, sistemele de securitate ar trebui proiectate ținând seama de acest fapt.

Adversar	Scop
Student	Pentru a se distra furând poșta electronică a celorlalți
Spărgător	Pentru a testa securitatea sistemului cuiva; pentru a fura date
Responsabil de vânzări	Pentru a pretinde că reprezintă toată Europa, nu numai Andorra
Om de afaceri	Pentru a descoperi planul strategic de marketing al competitorului
Fost funcționar	Pentru a se răzbuna că a fost concediat
Contabil	Pentru a sustrage bani de la o companie
Agent de vânzări	Pentru a nega o promisiune făcută clientului prin poștă electronică
Șarlatan	Pentru a fura numere de cărți de credit și a le vinde
Spion	Pentru a afla puterea militară a inamicului sau secrete industriale
Terorist	Pentru a fura secrete legate de conflicte armate

Fig. 8-1. Câteva persoane ce generează probleme de securitate și motivele acestora.

Problemele securității rețelei pot fi împărțite, în mare, în patru domenii strâns interconectate: confidențialitate, autentificare, nerepudiere și controlul integrității. Păstrarea secretului, denumită de asemenea și confidențialitate, se referă la păstrarea informației departe de utilizatorii neautorizați. Aceasta este ceea ce vine în mintea oamenilor atunci când se gândesc la securitatea rețelei. Autentificarea reprezintă determinarea identității persoanei cu care vorbești înainte de a dezvălui informații importante sau de a intra într-o afacere. Nerepudierea implică semnături: cum să dovedești că un client a făcut într-adevăr o comandă pentru zece milioane de nimicuri de 89 de cenți fiecare, dacă, mai târziu, el pretinde că prețul era de 69 de cenți? Sau poate susține că nu a făcut nici o comandă. În fine, cum poți fi sigur că un mesaj pe care l-ai primit a fost cel trimis cu adevărat și nu unul pe care un adversar răutăcios l-a modificat în tranzit sau l-a măsluit?

Toate aceste aspecte (confidențialitate, autentificare, nerepudiere și controlul integrității) apar și în sistemele tradiționale, dar cu câteva diferențe semnificative. Integritatea și confidențialitatea sunt realizate prin folosirea poștei înregistrate și prin sigilarea documentelor. Jefuirea trenului ce duce poșta este mai greu de realizat decât era în zilele lui Jesse James.

De asemenea, oamenii pot de obicei să spună ce diferență este între un document original și o fotocopie și adeseori numai primul are valoare pentru ei. Ca test, faceți o fotocopie a unui cec valid. Încercați luni să încasați de la bancă banii pe cecul original. Apoi încercați marți să încasați banii pe fotocopie. Observați diferența din comportamentul băncii. Cu cecuri electronice, originalul și copia nu sunt distinctibile. Va trece ceva vreme până când băncile vor ști cum să trateze astfel de situații.

Oamenii autentifică alți oameni prin recunoașterea fețelor, vocilor și scrisului lor. Dovada semnării se face prin semnături pe scrisori cu antet, sigilii etc. Falsificarea poate fi de obicei detectată prin scris, hârtie și experți în grafologie. Nici una din aceste opțiuni nu este disponibilă electronic. Evident, sunt necesare alte soluții.

Înainte de a intra în prezentarea acestor soluții, merită să consumăm câteva minute pentru a stabili unde anume în stiva de protocoale se situează securitatea rețelei. Probabil că nu există un singur loc. Fiecare nivel poate contribui cu ceva. La nivelul fizic, ascultarea firelor poate fi zădărnicită prin încapsularea liniilor de transmisie în tuburi sigilate conținând gaz de argon la presiuni înalte. Orice încercare de a perfora tubul va duce la pierderi de gaz, reducând presiunea și trăgând alarma. Câteva sisteme militare folosesc această tehnică.

La nivelul legătură de date, pachetele transmise pe o linie punct-la-punct pot fi codificate când părăsesc una dintre mașini și decodificate când intră în cealaltă. Toate detaliile pot fi manipulate la nivelul legătură de date, fără ca nivelurile mai înalte să aibă cunoștință de ceea ce se petrece. Aceas-

tă soluție eșuează, totuși, atunci când pachetele trebuie să traverseze mai multe rutere, deoarece pachetele trebuie decriptate în fiecare ruter, făcându-le astfel vulnerabile la atacurile din interiorul ruterelor. De asemenea, ea nu permite ca anumite sesiuni să fie protejate (de exemplu, acelea ce implică cumpărăturile on-line folosind cărți de credit), iar altele nu. Cu toate acestea, **criptarea legăturii** (eng.: link encryption), cum este numită această metodă, poate fi adăugată cu ușurință la orice rețea și este adeseori utilă.

La nivelul rețea, pot fi instalate ziduri de protecție pentru a păstra pachetele în interior sau pentru a păstra pachetele în afara acestuia. Securitatea IP funcționează de asemenea la acest nivel.

La nivelul transport, pot fi criptate conexiuni întregi, de la un capăt la celălalt, adică de la un proces la celălalt. Pentru o securitate maximă, este necesară securitatea capăt-la-capăt (eng.: end-to-end security).

În sfârșit, problemele cum sunt autentificarea utilizatorilor și nerepudierea nu pot fi tratate decât la nivelul aplicație.

Din moment ce securitatea nu se potrivește perfect cu nici un nivel, nu se potrivește în nici un capitol al acestei cărți. Din acest motiv, ea are propriul capitol.

Chiar dacă acest capitol este lung, tehnic și esențial, el este cvasi-irelevant pentru moment. Este bine știut faptul că cele mai multe probleme de securitate la bănci, de exemplu, se datorează angajaților incompetenți, procedurilor de securitate slabe, sau fraudelor interne mai degrabă decât unor criminali inteligenți care ascultă liniile telefonice și apoi decodează mesajele criptate. Dacă o persoană poate intra într-o filieră oarecare a unei bănci cu o hârtie ATM pe care a găsit-o pe stradă susținând că a uitat numărul de PIN și poate primi unul nou pe loc (în numele unor bune relații cu clienții), atunci toată criptografia din lume nu va preveni abuzurile. În această privință, cartea lui Roy Anderson vă ajută să „deschideți ochii”, deoarece documentează sute de exemple de eșecuri ale securității în numeroase industrii, aproape toate dintre ele datorându-se unor (ceea ce s-ar putea numi politicos) practici neglijente de afaceri sau lipsei de grijă pentru securitate (Anderson, 2001). Cu toate acestea, noi suntem optimiști că odată cu extinderea comerțului electronic (eng.: e-commerce), companiile își vor îmbunătăți procedurile operaționale, eliminând această fisură și aducând din nou aspectele de securitate în centrul atenției.

Exceptând securitatea de la nivelul fizic, aproape toată securitatea se bazează pe principii criptografice. Din acest motiv, vom începe studiul nostru asupra securității prin examinarea detaliată a criptografiei. În secțiunea 8.1, vom studia câteva dintre principiile de bază. Între secțiunile 8-2 și 8-5 vom examina câțiva dintre algoritmii și structurile de date fundamentale folosite în criptografie. Apoi vom examina în detaliu cum pot fi folosite aceste concepte pentru a obține securitatea în rețele. Vom concluziona cu câteva scurte gânduri despre tehnologie și societate.

Înainte de a începe, se impune o ultimă observație: ce nu este acoperit. Am încercat să ne concentrăm asupra problemelor din rețea, mai degrabă decât asupra problemelor de sisteme de operare sau aplicații, chiar dacă este deseori greu de tras o linie de demarcație. De exemplu, nu am spus nimic despre autentificarea utilizatorilor folosind biometria, securitatea parolelor, atacurile prin inundarea tampoanelor de memorie (eng. : buffer overflow), cai Troieni, falsificarea login-ului, bombe logice, viruși, viermi și altele. Toate aceste subiecte sunt acoperite în detaliu în Cap. 9 din *Sisteme de operare moderne* (Tanenbaum, 2001). Cititorul interesat de aspectele securității sistemelor este îndrumat către această carte. Acum haideți să începem călătoria noastră.

8.1 CRIPTOGRAFIA

Criptografie provine din cuvintele grecești pentru „scriere secretă". Criptografia are o istorie lungă și pitorească ce datează cu mii de ani în urmă. În această secțiune, vom schița doar câteva dintre aspecte, ca informații de bază pentru ceea ce urmează. Pentru o istorie completă este recomandată cartea lui Kahn (1995). Pentru o tratare detaliată a situației actuale în securitate și algoritmi, protocoale și aplicații criptografice, a se vedea (Kaufman et. al 2002). Pentru o abordare mai matematizată, a se vedea (Stinson, 2002). Pentru o abordare mai puțin matematică, a se vedea (Burnett și Paine, 2001).

Profesioniștii fac o distincție între cifruri și coduri. Un **cifru** este o transformare caracter-cu-caracter sau bit-cu-bit a mesajului, fără a ține cont de structura lingvistică a acestuia. Prin contrast, un **cod** înlocuiește un cuvânt cu un alt cuvânt sau cu un simbol. Codurile nu mai sunt folosite, deși au avut o istorie glorioasă. Cel mai reușit cod inventat vreodată a fost folosit de forțele armate ale S.U.A. în timpul celui de-al doilea război mondial în Pacific. Pur și simplu au pus indieni Navajo să vorbească între ei folosind cuvinte specifice dialectului Navajo pentru termenii militari, de exemplu *chay-da-gahi-nail-tsaidi* (literal: ucigaș de broaște țestoase) pentru armele anti-tanc. Limbajul Navajo este foarte tonal, extrem de complex și nu are o formă scrisă. Și nimeni din Japonia nu știa nimic despre el.

În septembrie 1945, *San Diego Union* descria codul spunând: „Pentru trei ani, oriunde acostau trupele Marine, Japonezii auzeau o mână de zgomote bolborosite răspândite printre alte sunete care semănau cu strigătul unui călugăr Tibetan și cu sunetul golirii unei sticle cu apă fierbinte. Japonezii nu au spart niciodată codul și mulți vorbitori de cod Navajo au fost răsplătiți cu onoruri militare înalte pentru servicii și curaj extraordinare. Faptul că S.U.A. a spart codul japonez, dar că japonezii nu au reușit niciodată să spargă codul Navajo a jucat un rol crucial în victoriile americane din Pacific.

8.1.1 Introducere în criptografie

Din punct de vedere istoric, patru grupuri de oameni au contribuit și au folosit arta criptografiei: armata, corpurile diplomatice, cei ce au ținut jurnale și îndrăgostiții. Dintre acestea, armata a avut rolul cel mai important și a dat contur domeniului de-a lungul secolelor. În interiorul organizațiilor militare, mesajele ce trebuiau criptate erau de obicei date pentru criptare și transmitere unor funcționari codori de nivel scăzut, prost plătiți. Volumul de mesaje nu permitea ca această muncă să fie făcută doar de câțiva specialiști de elită.

Până la apariția calculatoarelor, una din marile constrângeri ale criptografiei a fost capacitatea funcționarilor codori de a realiza transformările necesare, adeseori pe câmpul de luptă, cu echipament redus. O constrângere suplimentară a fost dificultatea de comutare rapidă de la o metodă criptografică la alta, deoarece aceasta implica reantrenarea unui număr mare de oameni. Totuși, pericolul ca un funcționar codor să fie capturat de către inamic a făcut să devină esențială posibilitatea de a schimba metoda criptografică instantaneu, în caz de necesitate. Aceste cerințe antagoniste au dat naștere modelului din fig. 8-2.

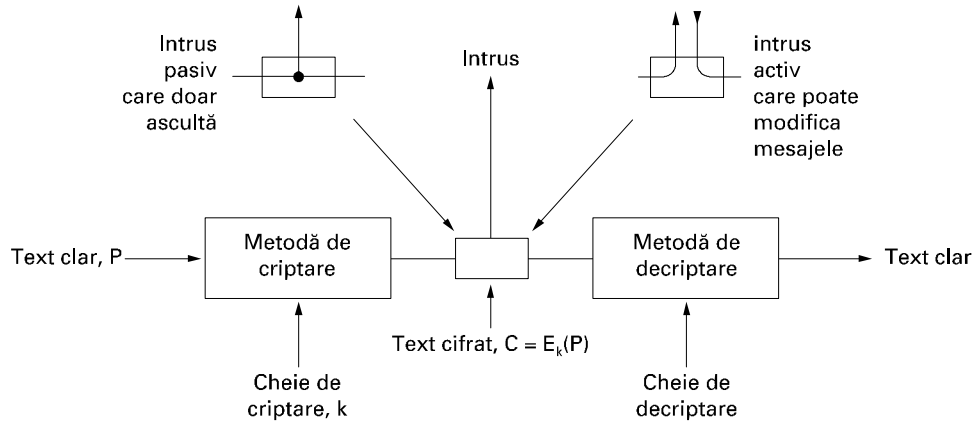


Fig. 8-2. Modelul de criptare (pentru un cifru cu chei simetrice).

Mesajele ce trebuie criptate, cunoscute sub numele de **text clar** (eng.: plain text), sunt transformate printr-o funcție parametrizată de o **cheie** (eng.: *key*). Rezultatul procesului de criptare, cunoscut sub numele de **text cifrat** (eng.: ciphertext), este apoi transmis, adeseori prin curier sau radio. Presupunem că inamicul, sau **intrusul**, ascultă și copiază cu acuratețe tot textul cifrat. Totuși, spre deosebire de destinatarul la care trebuie să ajungă, el nu știe care este cheia de decriptare și astfel nu poate decripta prea ușor textul cifrat. Uneori intrusul poate nu numai să asculte canalul de comunicație (intrus pasiv), ci și să înregistreze mesajele și să le retransmită mai târziu, să injecteze propriile sale mesaje sau să modifice mesajele legitime înainte ca ele să fi fost preluate de receptor (intrus activ). Arta de a sparge cifruri se numește **criptanaliză** (eng.: cryptanalysis). Arta de a sparge cifruri, denumită **criptanaliză** și arta de a le inventa (criptografia) sunt cunoscute împreună sub numele de **criptologie** (eng.: cryptology).

Adesea va fi util să avem o notație pentru a pune în relație textul clar, textul cifrat și cheile. Vom folosi $C=E_K(P)$ pentru a simboliza faptul că prin criptarea textului clar P, folosind cheia K, rezultă textul cifrat C. Similar, $P=D_K(C)$ reprezintă decriptarea lui C pentru a obține din nou textul clar. Rezultă că:

$$D_K(E_K(P))=P$$

Această notație sugerează că E și D sunt doar niște funcții matematice, ceea ce sunt de altfel. Singura parte mai delicată este aceea că ambele sunt funcții cu doi parametrii iar noi am scris unul dintre parametri (cheia) ca indice, nu ca argument, pentru a face distincție între ea și mesaj.

O regulă fundamentală a criptografiei este aceea că trebuie presupusă cunoașterea de către orice criptanalist a metodelor utilizate pentru criptare și decriptare. Cu alte cuvinte, criptanalistul știe cum lucrează în detaliu metoda de criptare E și de decriptare D din fig. 8-2. Cantitatea de efort necesară pentru a inventa, testa și instala o metodă nouă, ori de câte ori vechea metodă este compromisă (sau este presupus a fi compromisă), a făcut întotdeauna nepractică păstrarea secretă a algoritmului de criptare. A crede că este secretă atunci când nu este face mai mult rău decât bine.

Aici intră în scenă cheia. Cheia constă dintr-un șir (relativ) scurt care selectează una dintre mai multe criptări posibile. În contrast cu metoda generală, care poate fi schimbată doar o dată la câțiva ani, cheia poate fi schimbată oricât de des este nevoie. Astfel modelul nostru de bază este stabil și metoda generală, cunoscută de toată lumea, este parametrizată cu o cheie secretă și ușor de schim-

bat. Ideea că algoritmii sunt cunoscuți de criptanalist și că secretul constă exclusiv în cheie se numește **principiul lui Kerckhoff¸** denumit astfel după criptograful militar flamand Auguste Kerckhoff care l-a enunțat prima oară în 1883 (Kerckhoff, 1883). Astfel că avem:

Principiul lui Kerckhoff: Toți algoritmii trebuie să fie publici; numai cheile sunt secrete.

Caracterul nesecret al algoritmului nu poate fi subliniat suficient. Încercarea de a ține secret algoritmul, cunoscută în domeniu ca **securitate prin obscuritate**, nu funcționează niciodată. De asemenea, prin publicarea algoritmului, criptograful primește consultanță gratuită de la un număr mare de criptologi din mediul academic, nerăbdători să spargă sistemul pentru a putea publica articole care să demonstreze cât de inteligenți sunt ei. Dacă mulți experți au încercat să spargă algoritmul timp de 5 ani după publicarea lui și nici unul nu a reușit, probabil că algoritmul este destul de solid.

Din moment ce adevăratul secret este cheia, lungimea sa reprezintă un aspect foarte important de proiectare. Să considerăm o simplă combinație de seif. Principiul general este că se introduc cifre în secvență. Oricine știe aceasta, dar cheia este secretă. O lungime a cheii de două cifre înseamnă că există 100 de posibilități. O lungime a cheii de trei cifre înseamnă 1000 de posibilități și o lungime a cheii de șase cifre înseamnă un milion. Cu cât cheia este mai lungă, cu atât este mai mare **volumul de muncă** (eng.: work factor) pe care trebuie să-l depună criptanalistul. Volumul de muncă pentru a sparge sistemul prin căutare exhaustivă în spațiul cheilor este exponențial în raport cu lungimea cheii. Secretul provine din a avea un algoritm puternic (dar public) și o cheie lungă. Pentru a-l împiedica pe fratele tău mai mic să-ți citească poșta electronică, sunt suficiente chei de 64 de biți. Pentru folosirea uzuală în comerț, trebuie folosiți cel puțin 128 de biți. Pentru a păstra la distanță principalele guverne, sunt necesare chei de cel puțin 256 de biți, preferabil mai mulți.

Din punctul de vedere al criptanalistului, problema criptanalizei are trei variațiuni principale. Când are la dispoziție o cantitate de text cifrat și nici un fel de text clar, el este confruntat cu **problema textului cifrat** (eng.: ciphertext only problem). Criptogramele care apar la secțiunea de enigme a ziarelor pun acest tip de problemă. Când criptanalistul are câtva text clar și textul criptat corespunzător, problema este cunoscută sub numele de **problema textului clar cunoscut** (eng.: known plaintext problem). În sfârșit, atunci când criptanalistul are abilitatea de a cripta bucăți de text clar la propria sa alegere, avem de-a face cu **problema textului clar ales** (eng.: chosen plaintext problem). Criptogramele din ziare ar putea fi sparte trivial dacă criptanalistului i s-ar permite să pună întrebări de genul: Care este criptarea pentru ABCDEFGHIJKL?

Novicii în domeniul criptografiei presupun adeseori că dacă un cifru poate rezista unui atac de tip text cifrat (eng.: ciphertext only attack), el este sigur. Această presupunere este foarte naivă. În multe cazuri criptanalistul poate ghici corect unele părți din textul clar. De exemplu, primul lucru pe care multe sisteme îl afișează atunci când sunt accesate este login: . Echipat cu câteva perechi de text clar - text criptat potrivit, sarcina criptanalistului devine mult mai ușoară. Pentru a obține securitatea, criptograful trebuie să fie prudent și să se asigure că sistemul este rezistent, chiar dacă inamicul său poate cripta cantități arbitrare de text clar ales.

Metodele de criptare au fost istoric împărțite în două categorii: cifruri cu substituție și cifruri cu transpoziție. Vom studia acum pe scurt pe fiecare dintre aceste cifruri, ca informație fundamentală pentru înțelegerea criptografiei moderne.

8.1.2 Cifrurile cu substituție

Într-un **cifru cu substituție** fiecare literă sau grup de litere este înlocuit pentru a-l deghiza, cu altă literă sau grup de litere. Unul dintre cele mai vechi cifruri cunoscute este **Cifrul lui Caesar**, atribuit lui Julius Caesar. În această metodă, *a* devine *D*, *b* devine *E*, *c* devine *F*, ..., *z* devine *C*. De exemplu, cuvântul *attack* devine *DWWDFN*. În exemple textul clar va fi scris cu litere mici, iar textul cifrat va fi scris cu majuscule.

O ușoară generalizare a cifrului lui Caesar permite alfabetului textului cifrat să fie deplasat cu *k* litere, în loc de a fi deplasat întotdeauna cu 3. În acest caz, *k* devine o cheie pentru metoda generală a alfabetelor deplasate circular. Cifrul Caesar este posibil să îl fi păcălit pe Pompei, dar de atunci el nu mai păcălește pe nimeni.

Următoarea îmbunătățire este de a stabili o corespondență pentru fiecare simbol din textul clar, pentru simplitate să spunem cele 26 de litere, cu o altă literă. De exemplu:

textul clar:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	y	u	v	w	x	y	z
textul cifrat:	Q	W	E	R	T	Y	U	I	O	P	A	S	D	F	G	H	J	K	L	Z	X	C	V	B	N	M

Sistemul general de substituire simbol-cu-simbol este denumit **substituție monoalfabetică** (eng.: monoalphabetic substitution), cheia fiind șirul de 26 de litere corespunzând întregului alfabet. Pentru cheia anterioară, textul clar *attack* va fi transformat în texul cifrat *QZZQEA*.

La prima vedere, acesta ar putea fi considerat un sistem sigur deoarece, deși criptanalistul cunoaște sistemul general (substituție literă cu literă), el nu cunoaște care dintre cele 26! $\approx 4 \times 10^{26}$ chei posibile este folosită. Spre deosebire de cifrul lui Caesar, încercarea tuturor cheilor nu este o abordare prea promițătoare. Chiar și la 1 ns per soluție, unui calculator i-ar trebui 10^{10} ani pentru a încerca toate cheile.

Totuși, fiind dată o cantitate surprinzător de mică de text cifrat, cifrul poate fi spart cu ușurință. Atacurile de bază folosesc ca informație proprietățile statistice ale limbajelor naturale. În engleză, de exemplu, *e* este cea mai frecvent folosită literă, urmată de *t*, *o*, *a*, *n*, *i* etc. Cele mai comune combinații de două litere, sau **digrame** (eng.: *digrams*), sunt *th*, *in*, *er*, *re* și *an*. Cele mai comune combinații de trei litere, sau **trigrame** (eng.: *trigrams*), sunt *the*, *ing*, *and* și *ion*.

Un criptanalist care încearcă să spargă un cifru monoalfabetic va începe prin a număra frecvențele relative ale tuturor literelor din textul cifrat. După aceea el trebuie să încerce să asocieze cea mai frecventă literă cu *e*, următoarea cu *t*. Apoi el va căuta trigramele, pentru a o găsi pe cea mai frecventă cu forma *tXe*, care indică cu tărie că *X* este *h*. Similar, dacă apare frecvent șablonul *thYt*, probabil că *Y* îl înlocuiește pe *a*. Cu această informație, el poate căuta aparițiile frecvente ale trigramelor de forma *aZW*, care sunt cel mai probabil, *and*. Prin astfel de presupuneri făcute asupra celor mai comune litere, digrame, trigrame și cu ceva cunoștințe despre șabloanele asemănătoare de vocale și consoane, criptanalistul construiește literă cu literă o variantă de text clar.

O altă abordare este aceea de a ghici un cuvânt sau o expresie probabilă. De exemplu, considerați următorul text cifrat provenind de la o firmă de contabilitate (împărțit în blocuri de câte cinci caractere):

CTBMN	BYCTC	BTJDS	QXBNS	GSTJC	BTSWX	CTQTZ	CQVUI
QJSGS	TJQZZ	MNQJS	VLNSZ	VSZJU	JDSTS	JQUUS	JUBXJ
DSKSU	JSNTK	BGAQJ	ZBGYQ	TLCTZ	BNYBN	QJSW	

Un cuvânt probabil într-un mesaj provenind de la o firmă de contabilitate este *financial* (rom.: financiar). Folosind propriile noastre cunoștințe, cum ar fi faptul că *financial* are o literă care se repetă (*i*), cu alte patru litere între aparițiile ei, vom căuta în textul cifrat litere repetate aflate la această distanță. Găsim 12 potriviri pe pozițiile 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 și 82. Totuși, doar două dintre acestea, 31 și 42 au următoarea literă (cea corespunzând lui *n* în textul clar) repetată în locul corespunzător. Din acestea două, doar 31 are un *a* corect poziționat, deci știm că *financial* începe la poziția 30. Din acest moment, deducerea cheii este simplă folosind statisticile de frecvență a literelor în textele englezești.

8.1.3 Cifrurile cu transpoziție

Cifrurile cu substituție păstrează ordinea simbolurilor din textul clar, dar le deghizează. Spre deosebire de acestea, **cifrurile cu transpoziție** (eng.: transposition ciphers) reordonează literele, dar nu le deghizează. Fig. 8-3 descrie un cifru cu transpoziție simplu, transpoziția pe coloane. Cifrul are drept cheie un cuvânt sau o expresie ce nu conține litere repetate. În acest exemplu cheia este MEGABUCK. Scopul cheii este să numeroteze coloanele, coloana 1 fiind sub litera din cheie cea mai apropiată de începutul alfabetului ș.a.m.d. Textul clar este scris orizontal, pe rânduri. Textul cifrat este citit pe coloane, începând cu coloana al cărui număr de sub literă este mai mic.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Text clar

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Text cifrat

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Fig. 8-3. Un cifru cu transpoziție.

Pentru a sparge un cifru cu transpoziție, criptanalistul trebuie mai întâi să fie sigur că are de-a face cu un cifru cu transpoziție. Analizând frecvența de apariție pentru *E*, *T*, *A*, *O*, *I*, *N* etc., este ușor de văzut dacă ele se încadrează în șablonul obișnuit pentru text clar. Dacă da, cifrul este sigur un cifru cu transpoziție, deoarece într-un astfel de cifru, fiecare literă este reprezentată de ea însăși, păstrând distribuția frecvențelor.

Următorul pas care trebuie făcut este să se emită o presupunere asupra numărului de coloane. În multe cazuri un cuvânt sau o expresie probabilă poate fi ghicită din contextul mesajului. De exemplu, să presupunem că un criptanalist bănuiește că expresia în text clar *milliondollars* apare pe undeva prin mesaj. Se observă că în urma împachetării acestei expresii, în textul cifrat apar digramele *MO*, *IL*, *LL*, *LA*, *IR* și *OS*. Litera *O* din textul cifrat urmează după litera *M* din același text (adică sunt vertical adiacente în coloana 4) deoarece ele sunt separate în expresia probabilă de o distanță egală cu lungimea cheii. Dacă a fost folosită o cheie de lungime 7, în locul acestora ar fi trebuit să apară digramele *MD*, *IO*, *LL*, *LL*, *IA*, *OR* și *NS*. De fapt, pentru fiecare lungime a cheii, în textul cifrat sunt produse seturi diferite de digrame. Prin urmărirea diferitelor posibilități, criptanalistul poate determina cu ușurință lungimea cheii.

Pasul care mai rămâne este ordonarea coloanelor. Când numărul de coloane, *k*, este mic, poate fi examinată fiecare dintre cele *k(k-1)* perechi de coloane pentru a vedea dacă diagrama de frecvențe se potrivește diagramei pentru textul clar în engleză. Perechea cu cea mai bună potrivire se presupune că este corect poziționată. Acum fiecare coloană ce a rămas este încercată a fi succesorul acestei perechi. Coloana pentru care frecvența digramelor și trigramelor se potrivește cel mai bine se presupune a fi corectă. Coloana precedentă este găsită în același mod. Se continuă întregul proces până când o ordine potențială este descoperită. Există șanse ca textul clar să devină recognoscibil din acest stadiu (de exemplu, dacă apare *milloin*, este clar ce erori există în el).

Anumite cifruri cu transpoziție acceptă un bloc de lungime fixă la intrare și produc un bloc de lungime fixă la ieșire. Aceste cifruri pot fi descrise complet dându-se doar o listă în care să se precizeze ordinea în care caracterele vor fi trimise la ieșire. De exemplu, cifrul din fig. 8-3 poate fi văzut ca un cifru bloc pe 64 de caractere. Ieșirea sa este 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. Cu alte cuvinte, cel de-al patrulea caracter de la intrare, *a*, este primul ce va fi trimis la ieșire, urmat de al doisprezecelea, *f,* ș.a.m.d.

8.1.4 Chei acoperitoare

Construirea unui cifru imposibil de spart este de fapt destul de simplă; tehnica este cunoscută de decenii. În primul rând alegeți un șir aleatoriu de biți pe post de cheie. Apoi convertiți textul clar într-un șir de biți, de exemplu folosind reprezentarea ASCII. În final, calculați XOR (SAU eXclusiv) între cele două șiruri, bit cu bit. Textul cifrat rezultat nu poate fi spart, deoarece pentru un eșantion suficient de mare de text cifrat, fiecare literă va apărea la fel de des, de asemenea și orice digramă sau trigramă. Această metodă, cunoscută sub numele de **metoda cheilor acoperitoare** (eng.: one-time pad) este imună la toate atacurile din prezent și din viitor, indiferent de puterea de calcul pe care o are la dispoziție intrusul. Motivul provine din teoria informației: pur și simplu nu există nici o informație în mesaj deoarece orice text clar posibil cu o lungime dată este la fel de probabil.

Un exemplu de folosire al cheilor acoperitoare este prezentat în fig. 8-4. Mai întâi, mesajul 1, „*I love you.*” (rom.: „Te iubesc”) este convertit în cod ASCII pe 7 biți. Apoi o cheie acoperitoare, cheia acoperitoare 1, este aleasă și combinată XOR cu mesajul pentru a obține textul cifrat. Un criptanalist ar putea să încerce toate cheile acoperitoare posibile pentru a vedea ce text clar rezultă pentru fiecare. De exemplu, cheia acoperitoare 2 din figură poate fi testată, rezultând textul în clar 2, „*Elvis lives*” (rom.: „Elvis trăiește”) care ar putea să fie sau să nu fie plauzibil (un subiect în afara domeniului acestei cărți). De fapt, pentru fiecare text clar de 11 caractere ASCII, există o cheie acoperitoare care îl generează. Asta înseamnă ceea vrem să spunem prin afirmația că nu există nici o informație în mesaj: se poate obține din el orice mesaj cu lungimea corespunzătoare.

Mesajul 1:	1001001	0100000	1101100	1101111	1110110	1100101	0100000	1111001	1101111	1110101	0101110
Cheia acoperitoare 1:	1010010	1001011	1110010	1010101	1010010	1100011	0001011	0101010	1010111	1100110	0101011
Textul cifrat:	0011011	1101011	0011110	0111010	0100100	0000110	0101011	1010011	0111000	0010011	0000101
Cheia acoperitoare 2:	1011110	0000111	1101000	1010011	1010111	0100110	1000111	0111010	1001110	1110110	1110110
Textul în clar 2:	1000101	1101100	1110110	1101001	1110011	0100000	1101100	1101001	1110110	1100101	1110011

Fig. 8-4. Folosirea unei chei acoperitoare pentru criptare și posibilitatea de a obține orice text posibil din textul cifrat prin folosirea altei chei acoperitoare.

Cheile acoperitoare sunt extraordinare în teorie, dar au numeroase dezavantaje în practică. Unul dintre ele este faptul că nu poate fi memorată cheia, astfel încât atât transmițătorul cât și receptorul trebuie să poarte cu ei o copie scrisă a acesteia. Dacă vreunul dintre ei este capturat, evident că existența cheilor scrise nu este de dorit. În plus, cantitatea totală de date care poate fi transmisă este limitată de dimensiunea cheii disponibile. Dacă spionul dă lovitura și descoperă o comoară de date, el nu va fi capabil să le transmită înapoi la cartierul general deoarece cheia a fost epuizată. O altă problemă este sensibilitatea metodei la pierderea sau inserarea de caractere. Dacă transmițătorul și receptorul pierd la un moment dat sincronizarea, toate datele de aici încolo vor apărea ca fiind eronate.

Odată cu apariția calculatoarelor, metoda cheilor acoperitoare poate deveni practică pentru anumite aplicații. Sursa unei chei poate fi un DVD special care conține câțiva gigabiți de informație și care, transportat într-o cutie de DVD pentru filme, precedați de câteva minute de imagini, nu ar fi nici măcar suspect. Desigur, la viteza rețelei de ordinul gigabiților, a trebui să introduci un nou DVD la fiecare 5 secunde poate deveni obositor. Iar DVD-urile trebuie duse personal de la transmițător la receptor înainte de transmiterea oricărui mesaj, ceea ce reduce mult valoarea lor practică.

Criptografia cuantică

Interesant este că ar putea exista o soluție la problema transmiterii cheilor acoperitoare prin rețea, și ea provine dintr-o sursă neașteptată: mecanica cuantică. Ideea este încă experimentală, dar testele inițiale sunt promițătoare. Dacă poate fi perfecționată și eficientizată, teoretic toată criptografia va fi făcută folosind chei acoperitoare dovedit fiind că acestea sunt sigure. În continuare vom explica pe scurt cum funcționează această metodă, **criptografia cuantică**. În particular, vom descrie un protocol care se numește BB84 după autorii săi și anul publicării (Bennet și Brassard, 1984).

O utilizatoare, Alice, dorește să stabilească o cheie acoperitoare cu un al doilea utilizator, Bob. Alice și Bob sunt denumiți protagoniști, caracterele principale în povestea noastră. De exemplu, Bob este un bancher cu care Alice ar dori să facă afaceri. Numele „Alice” și „Bob” au fost folosite pentru protagoniștii principali în aproape fiecare studiu și lucrare despre criptografie în ultima decadă. Criptografii iubesc tradiția. Dacă am fi folosit „Andy” și „Barbara” pentru protagoniștii principali, nimeni nu ar fi crezut nimic din acest capitol. Deci, așa să fie.

Dacă Alice și Bob ar putea stabili o cheie acoperitoare, ar putea să o folosească pentru a comunica sigur. Întrebarea este: Cum pot ei să o stabilească fără să schimbe între ei DVD-uri? Putem presupune că Alice și Bob sunt la capetele opuse ale unei fibre optice prin care ei pot trimite și primi pulsuri de lumină. Cu toate acestea, o intrusă agitată, Trudy, poate tăia fibra pentru a introduce un dispozitiv de ascultare activ. Trudy poate citi toți biții din ambele direcții. Ea poate de asemenea să trimită mesaje false în ambele direcții. Situația ar părea fără speranță pentru Alice și Bob, dar criptografia cuantică poate pune subiectul într-o nouă lumină.

Criptografia cuantică este bazată pe faptul că lumina circulă în mici pachete numite **fotoni**, care au niște proprietăți specifice. În plus, lumina poate fi polarizată prin trecerea ei printr-un filtru polarizator, un fapt bine cunoscut, atât de purtătorii de ochelari de soare cât și de fotografi. Dacă o rază de lumină (adică un flux de fotoni) trece printr-un filtru polarizator, toți fotonii care ies din el vor fi polarizați pe direcția axului filtrului (adică pe verticală). Dacă raza va fi trecută acum printr-un al doilea filtru polarizator, intensitatea luminii care iese din al doilea filtru va fi proporțională cu rădăcina cosinusului unghiului dintre cele două axe. Dacă axele sunt perpendiculare, nu va trece nici un foton. Nu contează orientarea absolută ale celor două filtre; contează doar unghiul dintre axele lor.

Pentru a genera o cheie acoperitoare, Alice are nevoie de două seturi de filtre polarizatoare. Primul set constă dintr-un filtru vertical și un filtru orizontal. Această alegere se numește **bază recti-**

liniară. O bază (la plural: baze) este doar un sistem de coordonate. Al doilea set de filtre este similar, cu excepția faptului că sunt rotite cu 45 de grade, astfel că un filtru este de la stânga jos la dreapta sus, iar celălalt filtru este de la stânga sus la dreapta jos. Această alegere se numește **bază diagonală**. Deci Alice are două baze, pe care le poate introduce în raza de lumină oricând dorește. În realitate, Alice nu are patru filtre separate, ci un cristal a cărui polarizare poate fi comutată electric cu o viteză foarte mare la oricare dintre cele patru direcții permise. Bob are același echipament ca și Alice. Faptul că Alice și Bob au fiecare câte două baze disponibile este esențial în criptografia cuantică.

Pentru fiecare bază, Alice desemnează o direcție ca 0 și cealaltă ca 1. În exemplul prezentat mai jos, vom presupune că ea alege direcția verticală ca 0 și cea orizontală ca fiind 1. Independent, ea alege de asemenea diagonala stânga-jos dreapta-sus ca 0 și stânga-sus dreapta-jos ca 1. Ea trimite aceste opțiuni lui Bob în text necriptat.

Acum Alice alege o cheie acoperitoare, bazată de exemplu pe un generator de numere aleatoare (un subiect foarte complex în sine). Ea o transferă bit cu bit lui Bob, alegând aleator una dintre cele două baze disponibile pentru fiecare bit. Pentru a trimite un bit, tunul de fotoni emite un foton polarizat corespunzător pentru baza pe care o folosește pentru bitul respectiv. De exemplu, ea ar putea să aleagă bazele diagonală, rectiliniară, rectiliniară, diagonală, rectiliniară, etc. Pentru a trimite cheia sa acoperitoare 1001110010100110 cu aceste baze, ea va trimite fotonii din fig. 8-5(a). Fiind date cheia acoperitoare și secvența bazelor, polarizarea folosită pentru fiecare bit este unic determinată. Biții trimiși câte un foton la un moment dat se numesc **qubiți.**

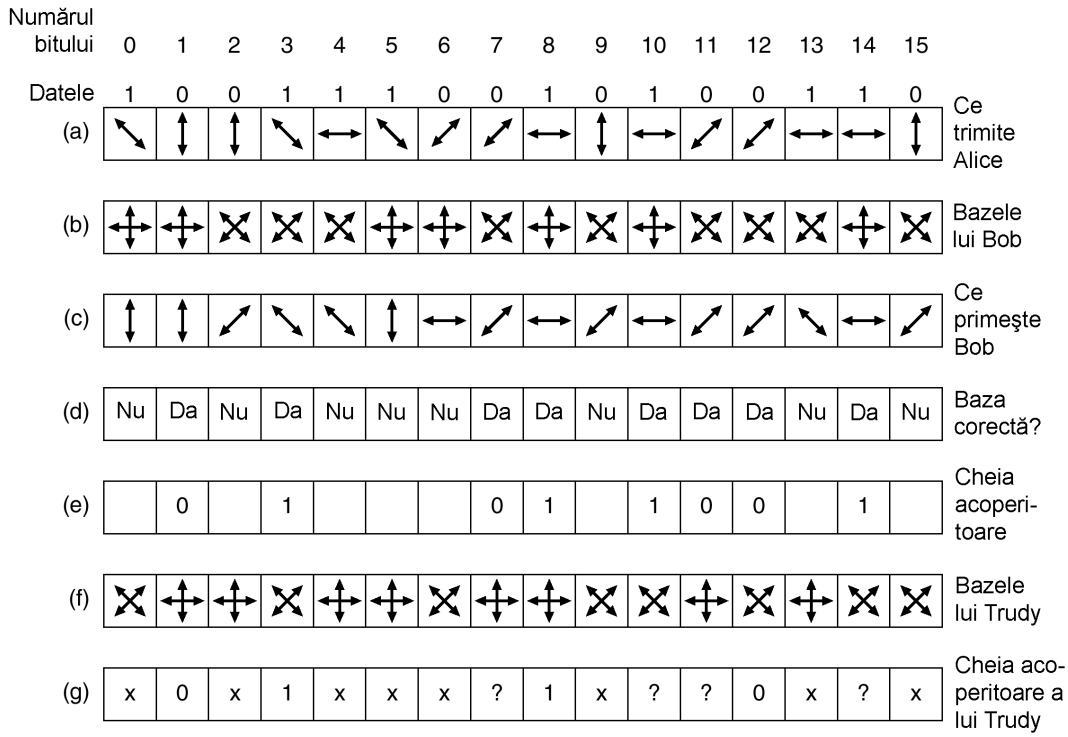

Numărul bitului
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Datele
1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0
(a)
Ce trimite Alice
(b)
Bazele lui Bob
(c)
Ce primeşte Bob
(d)
Nu Da Nu Da Nu Nu Nu Da Da Nu Da Da Da Nu Da Nu
Baza corectă?
(e)
0 1 0 1 1 0 0 1
Cheia acoperitoare
(f)
Bazele lui Trudy
(g)
x 0 x 1 x x x ? 1 x ? ? 0 x ? x
Cheia acoperitoare a lui Trudy

Fig. 8-5. Un exemplu de criptografie cuantică

Bob nu știe ce baze să folosească, prin urmare pentru fiecare foton sosit, alege o bază aleatoare și o folosește, după cum se poate vedea în fig. 8-5(b). Dacă alege baza corectă, el va obține bitul corect. Dacă alege baza incorectă, el obține un bit aleator deoarece dacă un foton trece printr-un filtru polarizat la 45 de grade față de polarizarea proprie, sare aleator la polarizarea filtrului sau sare cu o probabilitate egală, la o polarizare perpendiculară cu cea a filtrului. Această proprietate a fotonilor este fundamentală pentru mecanica cuantică. Prin urmare, câțiva dintre biți vor fi corecți iar câțiva vor fi aleatori, dar Bob nu știe care sunt unii și care sunt ceilalți. Rezultatele lui Bob sunt prezentate în fig. 8-5(c).

Dar cum află Bob ce baze a ghicit corect și pe care le-a greșit? Pur și simplu, el îi va spune lui Alice în text clar ce bază a folosit pentru fiecare bit, iar ea îi va spune care sunt corecte și care sunt greșite tot în clar, după cum este arătat în fig. 8-5(d). Din această informație, fiecare dintre ei poate construi un șir de biți din presupunerile corecte, după cum arată fig. 8-5(e). În medie, acest șir va fi jumătate din lungimea șirului inițial de biți, dar deoarece ambele părți îl cunosc, îl pot folosi pe post de cheie acoperitoare. Alice trebuie să trimită un șir puțin mai mare decât dublul lungimii dorite pentru ca ea și Bob să obțină o cheie acoperitoare cu lungimea dorită. Problemă rezolvată!

Dar stați un pic! Am uitat de Trudy. Să presupunem că ea este curioasă să știe ce are Alice de spus și taie fibra, inserând propriul ei detector și transmițător. Din nefericire pentru ea, nici ea nu știe ce bază să folosească pentru fiecare foton. Tot ce poate să facă este să aleagă aleatoriu o bază pentru fiecare foton, cum a făcut și Bob. Un exemplu cu alegerile ei este prezentat în fig. 8-5(f). Atunci când Bob raportează mai târziu (în text clar) ce baze a folosit și Alice îi spune (în text clar) care dintre ele sunt corecte, Trudy va ști ce a înțeles bine și ce nu a înțeles bine. În fig. 8-5 ea a nimerit bine biții 0, 1, 2, 3, 4, 6, 8, 12 și 13. Dar ea știe din replica lui Alice că doar biții 1, 3, 7, 8, 10, 11, 12 și 14 fac parte din cheia acoperitoare. Pentru patru dintre acești biți (1, 3, 8, 12) ea a ghicit corect și a capturat bitul corect . Ea nu a ghicit ceilalți patru (7, 10, 11 și 14) biți și nu cunoaște bitul transmis. Prin urmare, Bob știe că cheia acoperitoare începe cu 01011001, din fig. 8-5(e) dar tot ce are Trudy este 01?1??0?, din fig.8-5(g).

Bineînțeles că Alice și Bob sunt conștienți că Trudy ar fi putut captura o parte din cheia lor acoperitoare, astfel că ar dori să minimizeze informația deținută de Trudy. Ei pot face acest lucru prin efectuarea unei transformări asupra acestei chei. De exemplu, ei ar putea diviza cheia acoperitoare în blocuri de câte 1024 biți, ar putea ridica la pătrat fiecare bloc pentru a forma un număr de 2048 de biți și ar putea folosi concatenarea acestor numere de 2048 de biți ca cheie acoperitoare. Având cunoștințe parțiale despre șirul de biți transmis, Trudy nu are nici un mod de a genera pătratul lui și deci nu dispune de nici o informație. Transformarea din cheia acoperitoare originală într-una diferită care reduce cunoștințele lui Trudy se numește **amplificarea confidențialității**. În practică, în locul ridicării la pătrat sunt folosite transformări complexe în care fiecare bit de ieșire depinde de fiecare bit de intrare.

Biata Trudy. Nu numai că nu are nici o idee despre cheia acoperitoare, dar nici prezența ei nu este un secret. Ea trebuie să redirecteze fiecare bit recepționat de la Alice pentru Bob pentru a-l păcăli și a-l face să creadă că vorbește cu Alice. Problema este că tot ce poate ea să facă este să trimită qubitul pe care l-a recepționat, folosind polarizarea pe care a folosit-o pentru a-l recepționa și în jumătate din cazuri ea va greși, provocând multe erori în cheia acoperitoare a lui Bob.

Când în final Alice va începe să trimită datele, ea le va codifica folosind un puternic cod preventiv corector de erori. Din punctul lui Bob de vedere, o eroare de 1 bit în cheia acoperitoare este echivalentă cu o eroare de transmisie de 1 bit. În ambele cazuri, el va obține un bit eronat. Dacă există suficient cod preventiv corector de erori, el va putea recupera mesajul original în pofida tuturor erorilor, dar va putea foarte ușor să numere câte erori au fost corectate. Dacă acest număr este cu mult mai mare decât rata prevăzută de erori a echipamentului, el va ști că Trudy a interceptat linia și va

putea să acționeze în consecință (de ex., să-i spună lui Alice să comute pe un canal radio, să cheme poliția, etc.). Dacă Trudy ar dispune de o metodă de a clona un foton pentru a avea un foton pe care să-l inspecteze și un foton identic pe care să-l trimită lui Bob, ea ar putea evita detecția, dar în prezent nu este cunoscută nici o metodă de a clona perfect un foton. Dar chiar dacă Trudy ar putea clona fotoni, asta nu ar reduce valoarea criptografiei cuantice în stabilirea cheilor acoperitoare.

Deși a fost demonstrat că criptografia cuantică poate opera pe distanțe de 60 km de fibră, echipamentul este complex și scump. Totuși, ideea este promițătoare. Pentru mai multe informații despre criptografia cuantică, a se vedea (Mullins, 2002).

8.1.5 Două principii criptografice fundamentale

Deși în paginile ce urmează vom studia diferite sisteme criptografice, pentru toate acestea există două principii de bază a căror înțelegere este importantă.

Redundanța

Primul principiu este acela că toate mesajele criptate trebuie să conțină redundanță, adică informație ce nu este necesară pentru înțelegerea mesajului. Un exemplu poate clarifica de ce este nevoie de aceasta. Să considerăm o companie ce se ocupă cu comenzile prin poștă The Couch Potato (TCP), cu 60000 de produse. Crezând că vor fi foarte eficienți, programatorii de la TCP au decis că mesajele de comandă trebuie să conțină un nume de client pe 16 octeți, urmat de un câmp de date pe 3 octeți (1 octet pentru cantitate și 2 octeți pentru numărul produsului). Ultimii 3 octeți vor fi criptați folosind o cheie foarte lungă, cunoscută doar de client și de TCP.

La prima vedere sistemul pare sigur și, într-un anumit sens, chiar este, deoarece intrușii pasivi nu pot decripta mesajele. Din nefericire, există o slăbiciune fatală a acestui sistem, care îl face inutilizabil. Să presupunem că o funcționară recent concediată vrea să se răzbune pe TCP pentru că a dat-o afară. Chiar înainte de a pleca, ea ia lista clienților cu ea. Ea lucrează în timpul nopții și scrie un program care generează comenzi fictive folosind nume de clienți reali. Deoarece nu posedă lista cheilor, ea pune numere aleatorii în ultimii 3 octeți și trimite sute de comenzi la TCP.

Când sosesc aceste mesaje, calculatorul TCP folosește numele clientului pentru a localiza cheia și a decripta mesajul. Din nefericire pentru TCP, aproape fiecare mesaj de 3 octeți este valid, iar calculatorul începe să tipărească instrucțiunile trimise. Deși pare ciudat ca un client să comande 837 de seturi de leagăne pentru copii sau 540 de cutii cu nisip, calculatorul poate crede că acesta plănuiește să deschidă o mulțime de locuri de joacă. În acest mod, un intrus activ (ex-funcționara) poate cauza probleme imense, chiar dacă ea nu poate înțelege mesajele pe care le generează calculatorul ei.

Problema poate fi rezolvată prin adăugarea unor informații redundante tuturor mesajelor. De exemplu, dacă mesajele de comandă sunt extinse la 12 octeți, dintre care primii 9 trebuie să fie zerouri, atunci acest atac nu ar mai fi funcțional, deoarece ex-funcționara nu mai poate genera un șir mare de mesaje valide. Morala povestirii este aceea că toate mesajele trebuie să conțină o cantitate considerabilă de informație redundantă, astfel încât intrușii activi să nu poată trimite mesaje aleatorii care să fie interpretate ca mesaje valide.

Totuși, adăugarea informației redundante facilitează spargerea mesajelor de către criptanaliști. Să presupunem că afacerea de comenzi prin poștă este foarte competitivă și competitorul principal al companiei The Coach Potato, The Sofa Tuber, ar vrea tare mult să știe câte cutii de nisip vinde TCP. În consecință, ei ascultă linia telefonică a TCP. În schema originală, cu mesaje de 3 octeți, criptanaliza era aproape imposibilă, deoarece după ghicirea unei chei, criptanalistul nu avea cum să-și

dea seama dacă a ghicit corect. În fond, aproape orice mesaj este tehnic corect. Cu noua schemă de 12 octeți, este ușor pentru criptanalist să distingă un mesaj valid de unul invalid. Astfel că avem:

Principiul criptografic 1: Mesajele trebuie să conțină redundanță.

Cu alte cuvinte, după decriptarea unui mesaj, receptorul trebuie să poată distinge dacă este valid printr-o simplă inspectarea a acestuia și prin execuția unui calcul simplu. Această redundanță este necesară pentru a împiedica intrușii activi să înșele receptorul trimițându-i un mesaj fals și determinându-l să acționeze în numele mesajului decriptat. Cu toate acestea, aceeași redundanță facilitează intrușilor pasivi spargerea sistemului, deci aici apar unele probleme. Mai mult decât atât, redundanța nu trebuie niciodată să fie folosită sub forma a *n* zerouri la începutul sau sfârșitul unui mesaj, deoarece trecerea unor astfel de mesaje prin anumiți algoritmi criptografici dă rezultate predictibile, simplificând criptanaliza. Un cod CRC polinomial ar fi o alegere mult mai bună decât un șir de 0 deoarece receptorul îl poate verifica ușor, dar pentru criptanalist el reprezintă o muncă în plus. Și mai bună ar fi folosirea unei dispersii criptografice (eng.: cryptographic hash), un concept pe care-l vom explora mai târziu.

Să ne întoarcem un moment la criptografia cuantică și să vedem care este rolul redundanței acolo. Datorită interceptării fotonilor de către Trudy, câțiva dintre biții cheii acoperitoare a lui Bob vor fi eronați. Bob are nevoie de redundanță în mesajele primite pentru a determina faptul că există erori. O formă foarte primitivă de redundanță este transmiterea mesajului de două ori. Dacă cele două copii nu sunt identice, Bob știe că fie fibra optică are foarte multe zgomote, fie cineva interferează cu transmisia. Bineînțeles că trimiterea de două ori a fiecărui lucru este extrem de ineficient: o metodă mult mai eficientă de a detecta și corecta erorile este folosirea unui cod Hamming sau Reed-Solomon. Dar trebuie să fie clar că o anumită redundanță este necesară pentru a putea distinge între un mesaj valid și unul invalid, mai ales în prezența unui intrus activ.

Prospețimea

Cel de-al doilea principiu criptografic este acela că trebuie luate anumite măsuri pentru ne asigura că fiecare mesaj primit poate fi verificat că este proaspăt, adică a fost trimis foarte recent. Această măsură este necesară pentru a împiedica intrușii activi să retransmită mesaje mai vechi. Dacă nu se iau nici un fel de astfel de măsuri, ex-funcționara noastră ar putea asculta linia telefonică a TCP și ar putea retransmite mesajele valide trimise anterior. Reformulând această idee obținem:

Principiul criptografic 2: Este necesară o metodă pentru a dejuca atacurile prin replicarea mesajelor

O astfel de măsură este de a include în fiecare mesaj o amprentă de timp validă doar pentru, să spunem, 10 secunde. Receptorul trebuie doar să păstreze mesajele primite în ultimele 10 secunde, pentru a compara mesajele nou sosite cu anterioarele și pentru a filtra duplicatele. Mesajele mai vechi decât 10 secunde pot fi aruncate, deoarece orice replică a lor trimisă mai târziu de 10 secunde va fi refuzată ca fiind prea veche. Alte măsuri în afara amprentelor de timp vor fi discutate mai târziu.

8.2 ALGORITMI CU CHEIE SECRETĂ

Criptografia modernă utilizează aceleași idei de bază ca și criptografia tradițională (transpoziția și substituția) dar accentul este diferit. Tradițional, criptografii foloseau algoritmi simpli. În zilele

noastre este adevărat contrariul: obiectivul este de a face algoritmii de criptare atât de complecși și ireversibili, încât, chiar dacă un criptanalist achiziționează cantități imense de text cifrat la alegerea sa, el nu poate face nimic cu el fără a avea cheia.

Prima clasă de algoritmi de criptare pe care o vom studia în acest capitol se numesc **algoritmi cu cheie secretă** (eng.: symmetric-key algorithms) pentru că folosesc aceeași cheie pentru criptare și decriptare. Fig. 8-2 ilustrează folosirea unui algoritm cu cheie secretă. În particular, ne vom concentra asupra **cifrurilor bloc**, care primesc la intrare un bloc de n biți de text clar și îl transformă folosind cheia într-un bloc de text cifrat de n biți.

Algoritmii criptografici pot fi implementați fie în hardware (pentru viteză) sau în software (pentru flexibilitate). Deși cea mai mare parte a acestei lucrări tratează algoritmii și protocoalele, care sunt independente de implementarea efectivă, ar putea fi interesante câteva cuvinte despre construirea de hardware pentru criptare. Transpozițiile și substituțiile pot fi implementate cu circuite simple. Fig. 8-6(a) prezintă un dispozitiv, cunoscut sub numele de **cutie P** (P vine de la permutare), folosit pentru a efectua o transpoziție asupra unei intrări de 8 biți. Dacă cei 8 biți sunt selectați să fie notați de sus în jos cu 01234567, ieșirea acestei cutii P particulare este 36071245. Printr-o cablare internă corespunzătoare, o cutie P poate fi construită să efectueze orice transpoziție și să o facă practic cu viteza luminii.

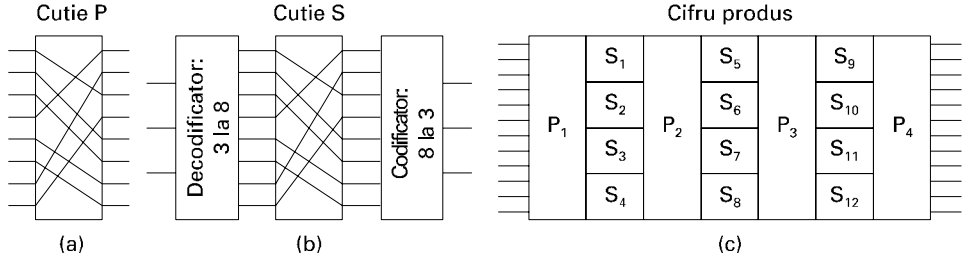


Fig. 8-6. Elemente de bază ale cifrurilor produs. (a) Cutie P. (b) Cutie S. (c) Produs.

Substituțiile sunt realizate de **cutiile S**, după cum este arătat în fig. 8-6(b). În acest exemplu este introdus un text clar de 3 biți, iar la ieșire este furnizat un text cifrat pe 3 biți. Intrarea de 3 biți selectează una dintre cele opt linii ce ies din primul nivel și o poziționează pe 1; toate celelalte linii sunt 0. Cel de-al doilea nivel este o cutie P. Ce de-al treilea nivel codifică din nou în binar linia selectată la intrare. Cu cablajul arătat, dacă opt numere scrise în octal 01234567 ar fi fost introduse unul după celălalt, secvența de ieșire ar fi 24506713. Cu alte cuvinte, 0 a fost înlocuit cu 2, 1 a fost înlocuit cu 4 etc. Din nou, prin cablarea corespunzătoare a cutiei P în interiorul cutiei S, poate fi realizată orice substituție. De altfel, un astfel de dispozitiv poate fi construit în hardware și poate atinge viteze foarte mari deoarece codificatorul și decodificatorul au doar una sau două întârzieri de porți logice (sub o nanosecundă) iar timpul de propagare prin cutia P poate fi mult mai mic decât o picosecundă.

Puterea efectivă a acestor elemente de bază devine vizibilă doar atunci când conectăm în cascadă o serie întreagă de cutii pentru a forma un **cifru produs**, după cum este arătat în fig. 8-6(c). În acest exemplu, 12 linii de intrare au fost transpuse (adică permutate) de primul nivel. Teoretic, ar fi posibil să avem la doilea nivel o cutie S care să pună în corespondență un număr de 12 biți cu alt număr de 12 biți. Totuși, un astfel de dispozitiv ar necesita 2^{12}=4096 cabluri încrucișate la nivelul său din mijloc. În schimb, intrarea este împărțită în patru grupuri de 3 biți, fiecare fiind substituit independent de celelalte. Cu toate că această metodă este mai puțin generală, ea este încă puternică. Prin inclu-

derea unui număr suficient de mare de niveluri în cifrul produs, ieșirea poate deveni o funcție extrem de complicată depinzând de intrare.

Cifrurile produs care operează asupra intrărilor de k biți pentru a produce ieșiri de k biți sunt destul de obișnuite. O valoare tipică pentru k este de la 64 la 256. O implementare hardware are de obicei cel puțin 18 niveluri fizice, nu doar șapte ca în fig. 8-6(c). O implementare software este programată ca o buclă cu cel puțin 8 iterații, fiecare dintre ele executând substituții ca cele ale cutiilor S pe sub-blocuri din blocurile de date cu dimensiuni de la 64 la 256 de biți, urmate o permutare care combină ieșirile cutiilor S. De obicei există o permutare inițială specială și, de asemenea, una la început. În literatură, iterațiile se numesc **runde**.

8.2.1 DES – Data Encryption Standard

În ianuarie 1977, guvernul SUA a adoptat ca standard oficial pentru informațiile nesecrete un cifru produs și dezvoltat de IBM. Acest cifru, **DES** (eng.: **Data Encryption Standard** – rom.: Standard pentru Criptarea Datelor), a fost adoptat extensiv în industrie pentru a fi utilizat în produsele de securitate. El nu mai este de mult sigur în forma sa originală, dar într-o formă modificată el este încă util. Vom explica acum cum lucrează DES.

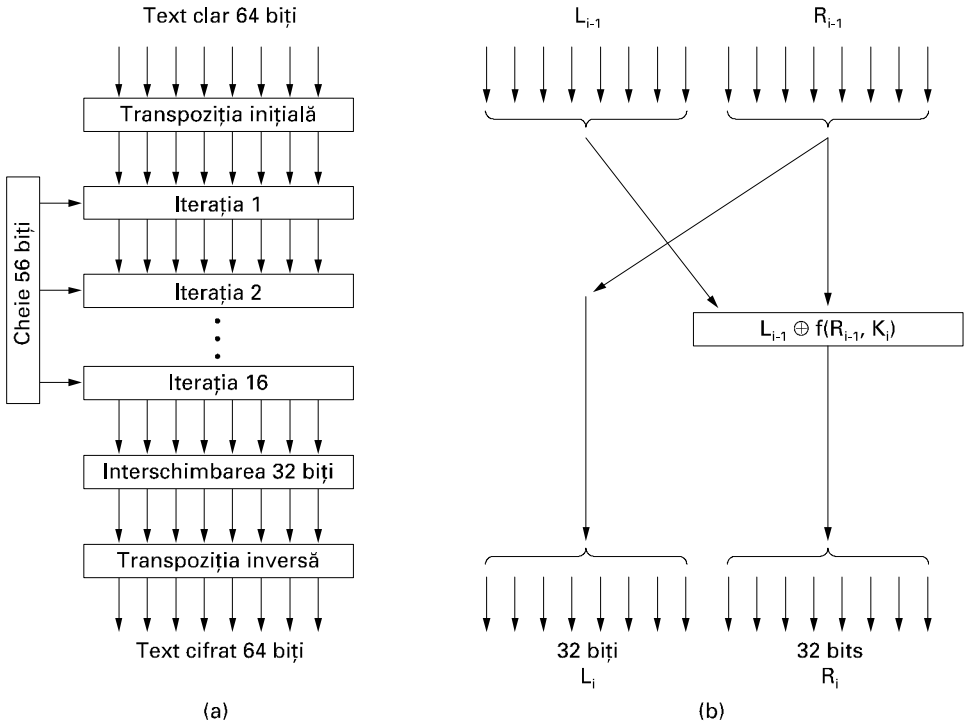


Fig. 8-7. Cifrul DES. (a) Schemă generală. (b) Detalierea unei iterații. Semnul + înconjurat de un cerc înseamnă SAU exclusiv

O prezentare generală a DES este făcută în fig. 8-7(a). Textul clar este criptat în blocuri de câte 64 de biți, rezultând blocuri de 64 de biți de text cifrat. Algoritmul, care este parametrizat cu o cheie de 56 de biți, are 19 runde distincte. Prima rundă este o transpoziție independentă de cheie, aplicată asupra textului clar de 64 de biți. Ultima rundă este exact inversa acestei transpoziții. Penultima rundă schimbă cei mai din stânga 32 de biți cu cei mai din dreapta 32 de biți. Cele 16 runde rămase sunt funcțional identice dar sunt parametrizate de funcții de cheie diferite. Algoritmul a fost proiectat pentru a permite ca decriptarea să se facă cu aceeași cheie ca și criptarea, o proprietate necesară în orice algoritm cu cheie secretă. Pașii sunt parcurși în ordine inversă.

Funcționarea unuia dintre pașii intermediari este ilustrată în fig. 8-7(b). Fiecare rundă ia două intrări de 32 de biți și produce două ieșiri de 32 de biți. Ieșirea din stânga este o simplă copie a intrării din dreapta. Ieșirea din dreapta rezultă în urma unui SAU exclusiv (XOR) bit cu bit între intrarea din stânga și o funcție depinzând de intrarea din dreapta și de o cheie pentru această rundă, K_i. Toată complexitatea rezidă în această funcție.

Funcția constă din patru pași, parcurși în secvență. În primul rând, este construit un număr de 48 de biți, E, prin expandarea celor 32 de biți ai lui R_{i-1} în concordanță cu o regulă de transpoziție fixă și de duplicare. În al doilea rând, E și K_i sunt combinate prin XOR. Ieșirea este apoi împărțită în opt grupuri de câte 6 biți și fiecare dintre acestea este introdus într-o cutie S diferită. Fiecare dintre cele 64 de intrări posibile într-o cutie S este pusă în corespondență cu o ieșire de 4 biți. În final, acești 8×4 biți sunt trecuți printr-o cutie P.

În fiecare din cele 16 iterații este folosită o cheie diferită. Înainte de începerea algoritmului este aplicată o transpoziție de 56 de biți asupra cheii. Chiar înainte de începerea fiecărei iterații, cheia este partiționată în două unități de câte 28 de biți, fiecare dintre ele este rotită la stânga cu un număr de biți depinzând de numărul iterației. K_i este derivat din această cheie rotită prin aplicarea unei transpoziții pe 56 de biți asupra ei. La fiecare rundă este extrasă și permutată o altă submulțime de 48 de biți din cei 56 de biți.

O tehnică folosită uneori pentru a face DES mai puternic se numește **albire** (eng.: whitening). Ea consistă în efectuarea operației SAU exclusiv între o cheie aleatoare de 64 de biți cu fiecare bloc de text clar înainte de a-l introduce în DES și apoi efectuarea încă a unui SAU exclusiv cu o a doua cheie de 64 de biți a textului cifrat înainte de a-l transmite. Albirea poate fi eliminată simplu prin efectuare operațiilor inverse (dacă receptorul are cele 2 chei de albire). Din moment ce această tehnică adaugă mai mulți biți la lungimea cheii, face căutarea exhaustivă a spațiului cheii mult mai mare consumatoare de timp. Se poate observa că aceeași cheie de albire este folosită pentru fiecare bloc (adică există o singură cheie de albire).

DES a fost învăluit în controverse încă din ziua în care a fost lansat. El se baza pe un cifru dezvoltat și brevetat de IBM, numit Lucifer, cu excepția faptului că cifrul IBM-ului folosea o cheie de 128 de biți în locul uneia de 56 de biți. Atunci când guvernul federal al SUA a dorit să standardizeze un cifru pentru folosire nesecretă, el a „invitat” IBM-ul să „discute” această problemă cu NSA, agenția spărgătoare de coduri a guvernului, care are ca angajați cel mai mare număr de matematicieni și criptologi din lume. NSA este atât de secretă, încât în industrie a apărut următoarea glumă:

Î: Ce înseamnă NSA?
R: No Such Agency (Nu există o astfel de agenție).

De fapt, NSA provine de la National Security Agency (rom.: Agenția Națională de Securitate).

După ce au avut loc aceste discuții, IBM a redus cheia de la 128 de biți la 56 de biți și a decis să păstreze secret procesul prin care a fost proiectat DES-ul. Mulți oameni suspectează faptul că lun-

gimea cheii a fost redusă pentru a exista siguranța că NSA poate sparge DES-ul, dar nici o organizație cu un buget mai mic nu ar putea să o facă. Păstrarea secretului proiectării a fost făcută probabil pentru a ascunde o trapă (ușă ascunsă) care ar putea ușura spargerea DES-ului de către NSA. Când un angajat al NSA a atenționat discret IEEE să abandoneze o conferința de criptografie planificată, acesta nu i-a făcut pe oameni să se simtă mai bine. NSA a negat totul.

În 1977, doi cercetători în criptografie de la Stanford, Diffie și Hellman (1977), au proiectat o mașină pentru a sparge DES-ul și s-a estimat că ea poate fi construită cu un buget de 20 de milioane de dolari. Dată fiind o mică bucată de text clar și textul cifrat corespunzător, această mașină ar putea să găsească cheia, prin căutarea exhaustivă a 2^{56} intrări din spațiul cheilor, în mai puțin de o zi. În prezent, o astfel de mașină ar costa mult sub 1 milion de dolari.

Triplu DES

Încă din 1979, IBM a realizat că lungimea cheii DES era prea mică și a conceput un mod de a o crește efectiv, folosind tripla criptare (Tuchman, 1979). Metoda aleasă, care de atunci a fost încorporată în Standardul Internațional 8732, este ilustrată în fig. 8-8. Aici sunt folosite două chei și trei runde. În prima rundă textul clar este în mod obișnuit criptat cu K_1. În a doua rundă, este rulat DES în mod de decriptare, folosind cheia K_2. În final, este efectuată o altă criptare cu K_1.

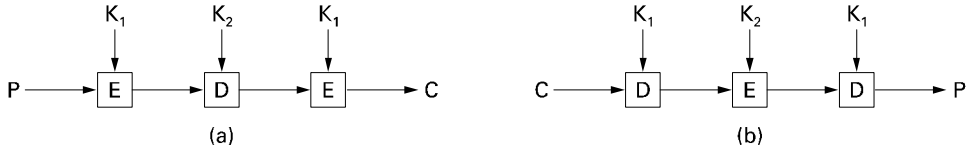


Fig. 8-8. (a)Tripla criptare folosind DES. (b) Decriptarea.

Această proiectarea dă naștere imediat la două întrebări. Prima, de ce sunt folosite doar două chei în loc de trei? A doua, de ce este folosită succesiunea de transformări **EDE** (eng.: **Encrypt Decrypt Encrypt**, rom.: Criptare Decriptare Criptare), în loc de **EEE** (eng.: **Encrypt Encrypt Encrypt**, rom.: Criptare Criptare Criptare)? Motivul pentru care sunt utilizate două chei este acela că chiar și cei mai paranoici criptografi admit că 112 biți sunt suficienți pentru aplicațiile comerciale de rutină, la momentul actual. A extinde la 168 de biți înseamnă a adăuga o supraîncărcare inutilă pentru gestiunea și transportul unei alte chei cu un câștig real redus.

Motivul pentru criptare, decriptare și apoi iar criptare este compatibilitatea cu produsele existente ce folosesc sisteme DES cu o singură cheie. Atât funcția de criptare cât și cea de decriptare sunt corespondențe între mulțimi de numere pe 64 de biți. Din punct de vedere criptografic, cele două corespondențe sunt la fel de puternice. Totuși, folosind EDE în locul EEE, un calculator ce utilizează tripla criptare poate comunica cu unul ce folosește criptarea simplă, folosind $K_1=K_2$. Această proprietate permite triplei criptări să fie pusă în practică treptat, lucru care nu interesează pe criptografii din mediul academic, dar care este de o importanță considerabilă pentru IBM și clienții săi.

8.2.2 AES – Advanced Encryption Standard

Deoarece DES a început să se apropie de sfârșitul utilei sale vieți , chiar și cu DES triplu, **NIST** (**National Institute of Standards and Technology**, rom: Institutul Național de Standarde și Tehnologie), agenția Departamentului de Comerț al Statelor Unite însărcinată cu aprobarea standardelor pentru Guvernul Federal al S.U.A., a decis că guvernul are nevoie de un nou standard criptografic

pentru folosință publică. NIST avea cunoștință de controversa din jurul DES și știa bine că dacă ar fi anunțat un nou standard orice persoană care știa câte ceva despre criptografie ar fi presupus în mod automat că NSA a construit o ușă secretă prin care NSA să poată citi orice criptat cu DES. În aceste condiții, probabil nimeni n-ar fi folosit standardul și acesta ar fi murit probabil singur.

De aceea NIST a avut o abordare surprinzător de diferită pentru o birocrație guvernamentală: a sponsorizat un concurs de criptografie. În ianuarie 1997, cercetători din toată lumea au fost invitați să depună propuneri pentru un nou standard, care urma să se numească **AES** (**Advanced Encryption Standard**, rom: Standard de Criptare Avansat). Regulile concursului erau:

1. Algoritmul trebuie să fie un cifru bloc simetric.
2. Tot proiectul trebuie sa fie public
3. Trebuie să fie suportate chei de 128, 192, și de 256 biți
4. Trebuie să fie posibile atât implementări hardware cât și software
5. Algoritmul trebuie să fie public sau oferit cu licență nediscriminatorie.

Au fost făcute cincisprezece propuneri serioase și au fost organizate conferințe publice în care propunerile au fost prezentate, iar publicul a fost încurajat să găsească punctele slabe în fiecare dintre ele. În august 1998, NIST a selectat cinci finaliști, în principal pe criterii de securitate, eficiență, flexibilitate și cerințe de memorie (importante pentru sistemele integrate). Au avut loc mai multe conferințe și s-au mai eliminat din variante. La ultima conferință s-a organizat un vot liber. Finaliștii și scorurile lor au fost următoarele:

1. Rijndael (din partea lui Joan Daemen și Vincent Rijmen, 86 voturi)
2. Serpent (din partea lui Ross Anderson, Eli Biham și Lars Knudsen, 59 voturi)
3. Twofish (din partea unei echipe condusă de Bruce Schneier, 31 voturi)
4. RC6 (din partea RSA Laboratories, 23 voturi)
5. MARS (din partea IBM, 13 voturi)

În octombrie 2000, NIST a anunțat că votează și el pentru Rijndael, iar în noiembrie 2001 Rijndael a devenit standard guvernamental al S.U.A. publicat ca Standard Federal de Procesare a Informațiilor nr.197 (Federal Information Processing Standard FIPS 197). Datorită deschiderii extraordinare a competiției, a proprietăților tehnice ale Rijndael și a faptului că echipa câștigătoare consta din doi tineri belgieni (despre care cu greu se poate presupune că au realizat și o ușă secretă doar pentru a face pe plac NSA), se așteaptă ca Rijndael să devină standardul criptografic dominant în lume pentru cel puțin o decadă. Numele Rijndael, pronunțat Rhine-doll (mai mult sau mai puțin), e derivat din numele autorilor: Rijmen + Daemen.

Rijndael permite lungimi de chei și mărimi de blocuri de la 128 de biți la 256 de biți în pași de câte 32 biți. Lungimea cheii și lungimea blocului pot fi alese în mod independent. Cu toate acestea, AES specifică faptul că mărimea blocului trebuie să fie de 128 de biți și lungimea cheii trebuie să fie de 128, 192, sau 256 de biți. E îndoielnic că cineva va folosi vreodată cheile de 192 de biți, astfel că de fapt, AES are două variante: bloc de 128 de biți cu cheie de 128 de biți și bloc de 128 de biți cu cheie de 256 de biți.

În prezentarea algoritmului, vom examina doar cazul 128/128 pentru că acesta va deveni cel mai probabil standardul comercial. O cheie de 128 de biți permite un spațiu al cheilor de $2^{128} \approx 2 \times 10^{38}$ chei. Chiar dacă NSA reușește să construiască o mașină cu un miliard de procesoare, fiecare fiind capabil să evalueze o cheie în fiecare picosecundă, ar trebui pentru o astfel de mașină aproximativ

10^{10} ani pentru a căuta în spațiul de chei. Până atunci soarele s-ar stinge, astfel că cei de atunci vor trebui să citească rezultatele la lumina lumânării.

Rijndael

Dintr-o perspectivă matematică, Rijndael se bazează pe Teoria Câmpului Galois, care îi conferă o serie de proprietăți de securitate demonstrabile. Cu toate acestea, poate fi privit și ca un cod C, fără a intra în explicații matematice.

Ca și DES, Rijndael folosește substituție și permutări, ca și runde multiple. Numărul de runde depinde de mărimea cheii și de mărimea blocului, fiind 10 pentru o cheie de 128 de biți cu blocuri de 128 biți și mărindu-se până la 14 pentru cheia cu cea mai mare dimensiune și blocul cu cea mai mare dimensiune. Totuși, spre deosebire de DES, toate operațiile sunt la nivel de octet, pentru a permite implementări eficiente hardware și software. Codul este dat în fig. 8-9.

```
#define LENGTH 16                                   /* # octeți în blocul de date sau în cheie */
#define NROWS 4                                     /* număr de linii din stare */
#define NCOLS 4                                     /* număr de coloane din stare */
#define ROUNDS 10                                   /* număr de iterații */
typedef unsigned char byte;                         /* întreg fără semn pe 8 biți */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                            /* index pentru iterare */
  byte state[NROWS][NCOLS];                         /* starea curentă */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];    /* cheile pentru runde */

  expand_key(key, rk);                              /* construiește cheile pentru runde */
  copy_plaintext_to_state(state, plaintext);        /* inițializează starea curentă */
  xor_roundkey_into_state(state, rk[0]);            /* face XOR între cheie și stare */

  for (r = 1; r <= ROUNDS; r++) {
    substitute(stare);                              /* aplică substituția fiecărui octet */
    rotate_rows(state);                             /* rotește rândul i cu i octeți */
    if (r < ROUNDS) mix_columns(state);             /* funcție de amestecare */
    xor_roundkey_into_state(state, rk[r]);          /* face XOR între cheie și stare */
  }
  copy_state_to_ciphertext(ciphertext, state);      /* întoarce rezultatul */
}
```

Fig. 8-9. Rijndael în linii generale

Funcția rijndael are trei parametri. Aceștia sunt: plaintext, un vector de 16 octeți conținând datele de intrare, ciphertext, un vector de 16 octeți în care va fi introdus rezultatul cifrat, și key, cheia de 16 octeți. Pe durata calculelor, starea curentă a datelor e păstrată într-un vector de octeți, state, a cărui mărime este NROWS × NCOLS. Pentru blocuri de 128 de octeți, acest vector este de 4 × 4 octeți. Întregul bloc de date de 128 de biți poate fi stocat în 16 octeți.

Vectorul state este inițializat cu textul clar și este modificat la fiecare pas al calculului. În anumiți pași, este realizată substituția octet-cu-octet. În alții, octeții sunt permutați în interiorul vectorului. Sunt folosite și alte transformări. La sfârșit, conținutul lui state reprezintă textul cifrat.

Codul începe prin expandarea cheii în 11 vectori de aceeași lungime cu starea. Aceștia sunt memorați în rk, care este un vector de structuri, fiecare structură conținând un vector stare. Unul dintre aceștia va fi folosit la începutul calculului și ceilalți 10 vor fi folosiți în timpul celor 10 runde, câte

unul în fiecare rundă. Calculul cheilor de runde din cheia de criptare este prea complicat pentru a fi prezentat aici. Este de ajuns să spunem că cheile de runde sunt produse prin rotiri repetate și aplicarea de operații XOR asupra unor grupuri de biți din cheie. Pentru toate detaliile, a se vedea (Daemen și Rijmen, 2002).

Următorul pas este acela de a copia textul clar în vectorul state astfel încât să poată fi procesat pe perioada rundelor. Acesta este copiat în ordinea coloanelor, cu primii patru octeți în coloana 0, următorii patru octeți în coloana 1 și așa mai departe. Atât coloanele cât și liniile sunt numerotate pornind de la 0, deși rundele sunt numerotate pornind de la 1. Setarea inițială a celor 12 vectori de octeți de dimensiuni 4×4 este ilustrată în fig. 8-10.

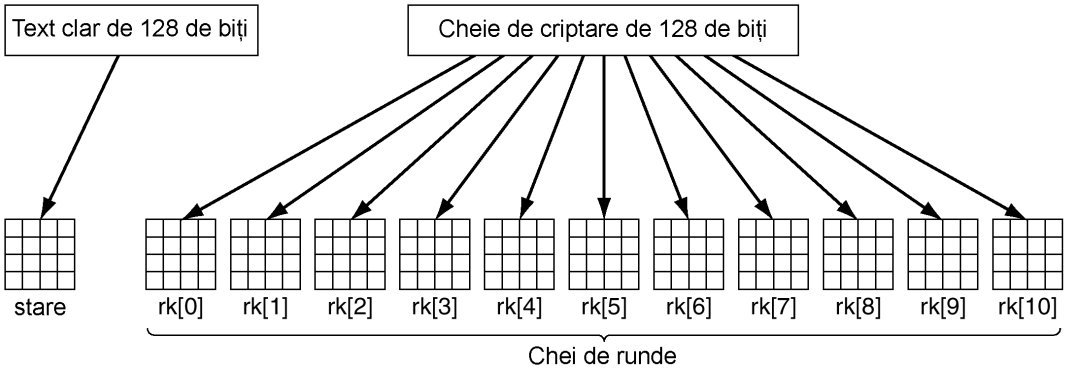


Fig. 8-10. Crearea vectorilor *state* și *rk*

Mai este un pas înainte de a începe calculul principal: rk[0] este combinat prin XOR în state, octet cu octet. Cu alte cuvinte fiecare octet din cei 16 aflați în state este înlocuit cu rezultatul aplicării operației XOR asupra sa și asupra octetului corespunzător din rk[0].

Și acum urmează partea cea mai interesantă. Bucla execută 10 iterații, câte una pe rundă, transformând state la fiecare iterație. Fiecare rundă constă în patru pași. Pasul 1 realizează o substituție octet-cu-octet asupra lui state. Pe rând, fiecare octet este folosit ca index într-o cutie S pentru a-i înlocui valoarea prin conținutul corespunzător din acea cutie S. Acest pas este un cifru de substituție monoalfabetică directă. Spre deosebire de DES, care are mai multe cutii S, Rijndaek are doar o cutie S.

Pasul 2 rotește la stânga fiecare din cele 4 rânduri. Rândul 0 este rotit cu 0 octeți (nu e schimbat), rândul 1 este rotit cu 1 octet, rândul 2 este rotit cu 2 octeți și rândul 3 este rotit cu 3 octeți. Acest pas difuzează conținutul datelor curente în jurul blocului, analog cu permutările din fig. 8-6.

Pasul 3 amestecă fiecare coloană independent de celelalte. Această amestecare este realizată prin înmulțire de matrice, în care noua coloană este produsul dintre vechea coloană și o matrice constantă, multiplicarea fiind realizată folosind câmpul finit Galois, $GF(2^8)$. Deși acest lucru poate părea complicat, există un algoritm care permite fiecărui element al noii coloane să fie calculat folosind două căutări în tabele și trei operații XOR (Daemen și Rijmen, 2002, Anexa E).

În fine, pasul 4 aplică operația XOR pentru cheia din runda curentă și vectorul stare.

Deoarece fiecare pas e reversibil, decriptarea poate fi realizată prin rularea algoritmului de la coadă la cap. Oricum, este posibilă și o schemă prin care decriptarea poate fi realizată prin rularea algoritmului de criptare nemodificat, dar folosind tabele diferite.

Algoritmul a fost proiectat nu doar pentru o securitate foarte solidă, dar și pentru o viteză foarte mare. O bună implementare software pe o mașină la 2 GHz ar trebui să atingă o rată de criptare de 700 Mbps, ceea ce este suficient de rapid pentru a criptarea peste 100 de fișiere video MPEG-2 în timp real. Implementările hardware sunt chiar mai rapide.

8.2.3 Moduri de cifrare

În ciuda acestei complexități, AES (sau DES sau orice cifru bloc de altfel) este de fapt un cifru de substituție monoalfabetică care folosește caractere „mari" (caractere de 128 de biți pentru AES și caractere de 64 de biți pentru DES). Ori de câte ori același bloc de text clar intră pe la un capăt, același text cifrat iese pe la celălalt. Dacă criptezi textul clar *abcdefgh* de 100 de ori cu aceeași cheie DES, obții același text cifrat de 100 de ori. Un intrus poate să exploateze această proprietate pentru a submina cifrul.

Modul cu carte de coduri electronică

Pentru a vedea cum poate fi folosită această proprietate a cifrului cu substituție monoalfabetică pentru a submina DES-ul, vom folosi (triplu) DES deoarece este mai ușor să prezinți blocuri de 64 de biți decât blocuri de 128 de biți; AES are însă exact aceeași problemă. Modul direct de a folosi DES pentru a cripta o bucată lungă de text clar este de a o sparge în blocuri consecutive de 8 octeți (64 de biți) și de a le cripta unul după altul cu aceeași cheie. Ultima bucată de text clar este completată până la 64 de biți dacă este nevoie. Tehnica este cunoscută ca modul ECB (Electronic Code Book, rom: modul cu carte de coduri electronică), analog cu cărțile de coduri electronice de modă veche unde era precizat fiecare cuvânt de text clar, urmat de textul său cifrat (de obicei un număr zecimal de 5 cifre).

În fig. 8-11 este prezentat începutul unui fișier conținând primele anuale ale unei companii care s-a decis să-și premieze angajații. Fișierul constă din înregistrări de 32 de octeți, câte o înregistrare pentru fiecare angajat, în formatul arătat: 16 octeți pentru nume, 8 octeți pentru funcția ocupată și 8 octeți pentru primă. Fiecare din cele 16 blocuri de 8 octeți (numerotate de la 0 la 15) este criptat cu (triplu) DES.

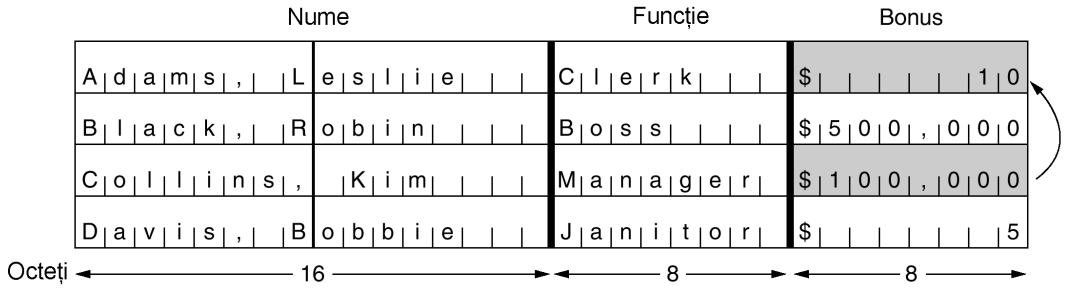


Fig. 8-11. Textul clar al unui fișier criptat ca 16 blocuri DES

Leslie a avut o controversă cu șeful și nu așteaptă prea mult de la această primă. În schimb Kim este favorita șefului și oricine știe asta. Leslie poate avea acces la fișier după ce el a fost criptat, dar înainte de a fi trimis la bancă. Poate Leslie să rectifice această situație nedreaptă, dat fiind doar fișierul criptat?

Nici o problemă. Tot ceea ce are Leslie de făcut este să realizeze o copie a celui de-al 12-lea bloc de text cifrat (care conține prima lui Kim) și să-l folosească pentru a înlocui blocul de text cifrat cu numărul 4 (care conține prima lui Leslie). Chiar și fără a ști ce cuprinde blocul 12, Leslie se poate aștepta să aibă un Crăciun mai fericit anul acesta. (Copierea blocului de text cifrat 8 este de asemenea o posibilitate, dar este mai probabil să fie descoperită; în plus, Leslie nu este o persoană lacomă).

Modul cu înlănțuirea blocurilor cifrate

Pentru a para acest tip de atac, toate cifrurile bloc trebuie înlănțuite în diferite moduri astfel încât înlocuirea unui bloc în modul în care a făcut-o Leslie să conducă la situația în care textul clar decriptat, începând cu blocul înlocuit, să fie nefolositor. Un mod de înlănțuire este înlănțuirea blocurilor cifrate (eng.: cipher block chaining). În această metodă, prezentată în fig. 8-12, fiecare bloc de text clar este combinat prin XOR cu blocul anterior de text cifrat, înainte de a fi criptat. În consecință, același bloc de text clar nu se va mai pune în corespondență cu același bloc de text cifrat, iar criptarea nu mai este o mare substituție monoalfabetică. Primul bloc este combinat prin XOR cu un vector de inițializare, IV (Initialization Vector), ales aleatoriu, care este transmis împreună cu textul cifrat.

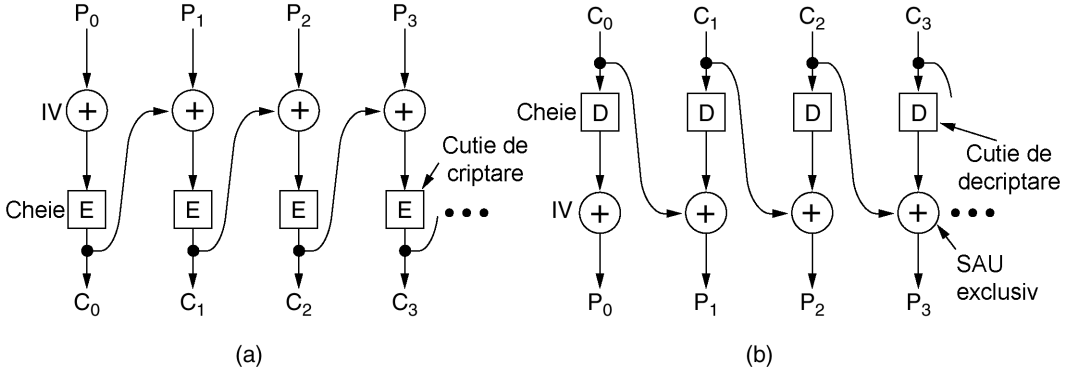


Fig. 8-12. Înlănțuirea blocurilor cifrate (Cipher block chaining). (a) Criptare. (b) Decriptare.

Putem vedea cum lucrează înlănțuirea blocurilor cifrate prin examinarea exemplului din fig. 8-12. Putem începe prin a calcula $C_0=E(P_0 \text{ XOR } IV)$. Apoi vom calcula $C_1=E(P_1 \text{ XOR } C_0)$ și așa mai departe. Decriptarea folosește tot XOR pentru a inversa procesul, cu $P_0=IV \text{ XOR } D(C_0)$, și așa mai departe. De notat că criptarea blocului i este o funcție de toate textele clare din blocurile de la 0 la i-1, astfel încât același text clar va genera text cifrat diferit în funcție de locul unde apare. O transformare de tipul celei făcute de Leslie va avea ca rezultat un nonsens în cele două blocuri ce încep din câmpul de primă al lui Leslie. Pentru un ofițer de securitate perspicace, această caracteristică poate sugera de la cine să pornească investigația. Înlănțuirea blocurilor cifrate are de asemenea avantajul că același bloc de text clar nu va rezulta în același bloc de text cifrat, făcând criptanaliza mai dificilă. De fapt, acesta este principalul motiv pentru care este folosită.

Modul cu reacție cifrată

Cu toate acestea, înlănțuirea blocurilor cifrate are dezavantajul că este necesar ca un întreg bloc de 64 de biți să sosească înainte ca decriptarea să poată începe. Acest mod este nepotrivit pentru folosirea în cazul terminalelor interactive, unde o persoană poate să introducă linii mai scurte de 8 caractere și apoi să se oprească în așteptarea unui răspuns. Pentru criptările octet-cu-octet poate fi utilizat **modul cu reacție cifrată** (eng.:**Cipher feedback mode**), folosind (triplu) DES, ca în fig. 8-13.

Pentru AES ideea este aceeași, numai că se folosește un registru de deplasare de 128 de biți. În această figură, este arătată starea mașinii de criptare după ce octeții 0 până la 9 au fost criptați și trimiși. Când sosește blocul 10 din textul clar, după cum este ilustrat în fig. 8-13(a), algoritmul DES operează asupra registrului de deplasare 64 de biți pentru a genera 64 de biți de text cifrat. Octetul cel mai din stânga al textului cifrat este extras și combinat prin XOR cu P_{10}. Acest octet este transmis pe linie. În plus, registrul de deplasare este deplasat cu 8 biți la stânga, provocând ieșirea lui C_2 pe la capătul din stânga și inserarea lui C_{10} în poziția care tocmai a rămas vacantă la dreapta lui C_9. De notat că conținutul registrului de deplasare depinde de întreaga istorie anterioară a textului clar, astfel încât un șablon care se repetă de mai multe ori în textul clar va fi criptat de fiecare dată diferit în textul cifrat. La fel ca la înlănțuirea blocurilor cifrate, este necesar un vector de inițializare pentru a porni rostogolirea mingii.

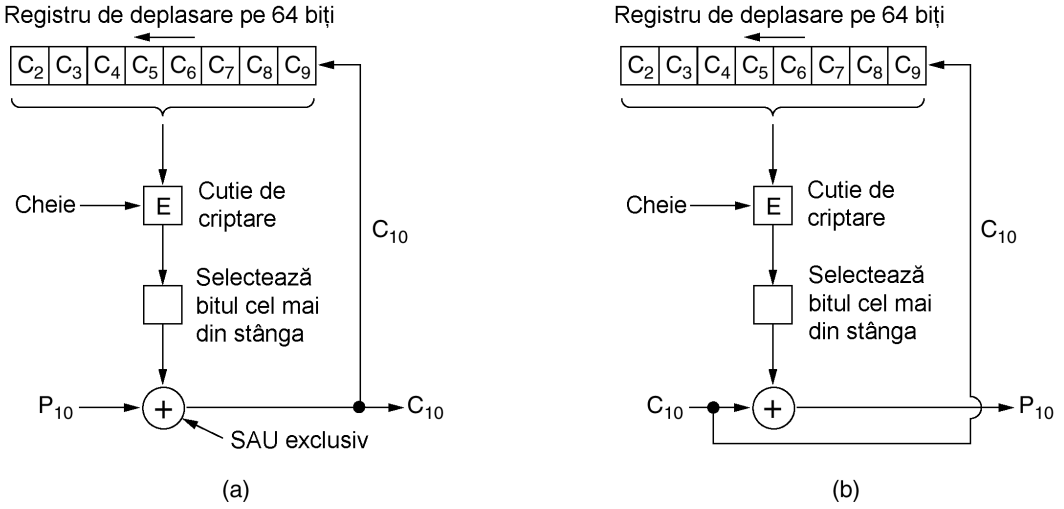


Fig. 8-13. Modul cu reacție cifrată (Cipher feedback mode). (a) Criptare. (b) Decriptare.

Decriptarea în modul cu reacție cifrată face același lucru ca și criptarea. În particular, conținutul registrului de deplasare este *criptat*, nu *decriptat*, astfel încât octetul selectat care este combinat prin XOR cu C_{10} pentru a obține P_{10} este același cu cel ce a fost combinat prin XOR cu P_{10} pentru a-l obține pe C_{10} prima dată. Atâta vreme cât cele două registre de deplasare rămân identice, decriptarea lucrează corect. Acest lucru este ilustrat în fig. 8-13(b).

O problemă a modului cu reacție cifrată este că, dacă un bit al textului cifrat este inversat accidental în timpul transmisiei, cei 8 octeți care sunt decriptați în timp ce octetul greșit este în registrul de deplasare vor fi corupți. Odată ce octetul greșit este împins afară din registrul de deplasare, se va genera din nou text clar corect. Astfel, efectele unui singur bit inversat sunt relativ localizate și nu distrug restul mesajului, dar distrug atâția biți câți are ca lățime registrul de deplasare.

Modul cu cifru înlănțuit

Cu toate acestea, există aplicații în care a avea o eroare de transmisie de 1 bit care să strice 64 de biți de text clar reprezintă o pierdere prea mare. Pentru aceste aplicații există o a patra opțiune, **modul cu cifru înlănțuit** (eng.: **Stream Cipher Mode**). Acesta lucrează criptând un vector de inițializare, folosind o cheie pentru a obține un bloc de ieșire. Blocul de ieșire este apoi criptat folosind cheia pentru a obține un al doilea bloc de ieșire. Acest bloc este apoi criptat pentru a obține un al treilea

bloc și așa mai departe. Secvența (arbitrar de mare) de blocuri de ieșire, numită **lanțul de chei** (eng.: **keystream**), este tratată ca cheie acoperitoare (eng.: one-time pad) și este combinată cu textul clar pentru a obține textul cifrat, ca în fig. 8-14(a). De notat că vectorul IV este folosit doar la primul pas. După aceasta, ieșirea este criptată. De notat de asemenea că lanțul de chei este independent de date, deci poate fi calculat în avans, dacă este nevoie, și este complet insensibil la erori de transmisie. Decriptarea este arătată în fig. 8-14(b).

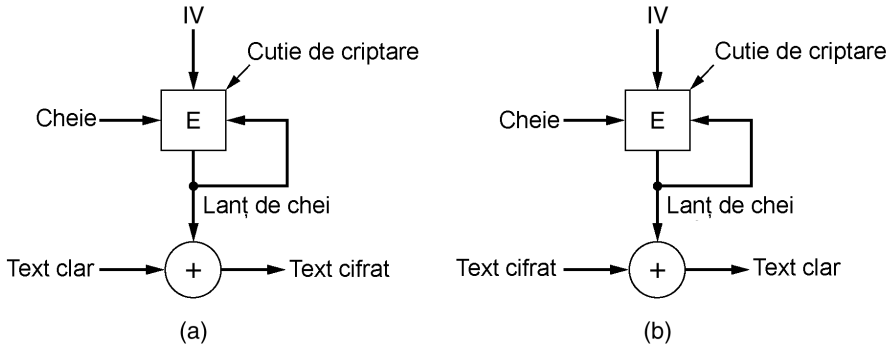


Fig. 8-14. Un cifru înlănțuit. (a) Criptare. (b) Decriptare.

Decriptarea se face generând același lanț de chei la receptor. Deoarece lanțul de chei depinde doar de IV și de cheie, el nu este afectat de erori de transmisie în textul cifrat. Astfel, o eroare de 1 bit în textul cifrat transmis generează doar o eroare de 1 bit în textul clar decriptat.

Este esențial să nu se folosească aceeași pereche (cheie, IV) de două ori cu un cifru înlănțuit deoarece acest lucru va genera același lanț de chei de fiecare dată. Folosirea aceluiași lanț de chei de două ori expune textul cifrat la un **atac prin refolosirea lanțului de chei** (eng.: **keystream reuse attack**). Imaginați-vă că blocul de text clar, P_0 este criptat cu lanțul de chei pentru a obține P_0 XOR K_0. Mai târziu, un al doilea bloc de text clar Q_0 este criptat cu același lanț de chei pentru a obține Q_0 XOR K_0. Un intrus care capturează ambele texte cifrate poate face simplu XOR între ele pentru a obține P_0 XOR Q_0, ceea ce elimină cheia. Intrusul are acum combinația XOR a celor două blocuri de text clar. Dacă unul dintre ele este știut sau poate fi ghicit, atunci și celălalt poate fi găsit. În orice caz, un XOR a două șiruri de text clar poate fi atacat folosind proprietățile statistice ale mesajului. De exemplu, pentru text în limba engleză, cel mai comun caracter din șir va fi probabil XOR între două spații, urmat de XOR între spațiu și litera "e". Pe scurt, echipat cu un XOR între două texte clare, criptanalistul are o șansă excelentă de a le deduce pe ambele.

Modul contor

O problemă pe care o au toate modurile în afară de modul cu carte de coduri electronică este că e imposibilitatea accesului aleator la datele cifrate. De exemplu, să presupunem că un fișier este transmis printr-o rețea și apoi este stocat pe disc în forma criptată. Acesta ar fi un procedeu rezonabil când calculatorul receptor este un notebook (rom.: carnet de notițe) care ar putea fi furat. Stocarea tuturor fișierelor critice în formă criptată reduce mult problemele datorate unor scurgeri de informații în eventualitatea în care calculatorul ajunge în mâini nepotrivite.

Cu toate acestea, fișierele de pe disc sunt deseori accesate în ordine aleatoare, în special fișierele din baze de date. Cu o criptare de fișiere care folosește înlănțuire de blocuri cifrate, accesul la un bloc aleator necesită întâi decriptarea tuturor blocurilor dinaintea lui, ceea ce reprezintă o soluție

costisitoare. Din acest motiv s-a inventat încă un mod, **modul contor** (eng.: **Counter Mode**), schițat în fig. 8-15. Aici textul clar nu este criptat direct. În schimb, se criptează vectorul de inițializare plus o constantă și textul cifrat rezultat se combină prin XOR cu textul clar. Crescând cu 1 vectorul de inițializare pentru fiecare bloc nou este ușor să decriptezi un bloc de oriunde din fișier fără a fi nevoie să îi decriptezi toți predecesorii.

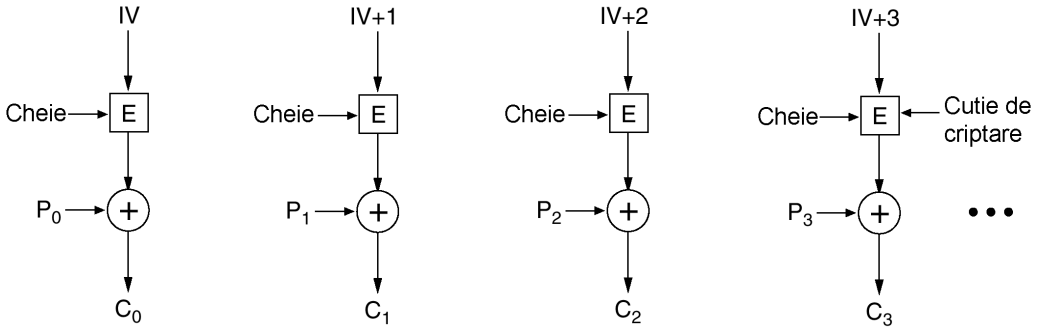


Fig. 8-15. Criptarea folosind modul contor

Deși modul contor este folositor, el are o slăbiciune care merită să fie evidențiată. Să presupunem că aceeași cheie, K, este folosită din nou ulterior (cu un text clar diferit dar cu același IV) și un atacator obține tot textul cifrat în ambele rulări. Lanțurile de chei sunt aceleași în ambele cazuri, expunând cifrul la un atac la refolosirea lanțului de chei similar celui de la cifrurile înlănțuite. Tot ce are criptanalistul de făcut este XOR între cele două texte cifrate pentru a elimina toată protecția criptografică și a obține combinația XOR între textele clare. Această slăbiciune nu înseamnă că modul contor este o idee proastă. Înseamnă doar că și cheile și vectorii de inițializare ar trebui alese independent și aleator. Chiar dacă aceeași cheie este folosită accidental de două ori, dacă vectorul IV este diferit de fiecare dată, textul clar este în siguranță.

8.2.4 Alte cifruri

DES și Rijndael sunt cei mai cunoscuți algoritmi criptografici cu cheie simetrică. Cu toate acestea merită să menționăm că au fost proiectate numeroase alte cifruri cu cheie simetrică. Câteva dintre acestea sunt integrate cu diverse produse. Câteva dintre cele mai comune sunt enumerate în fig. 8-16.

Cifru	Autor	Lungimea cheii	Comentarii
Blowfish	Bruce Schneier	1 - 448 de biți	vechi și lent
DES	IBM	56 de biți	prea slab pentru a mai fi folosit acum
IDEA	Massey și Xuejia	128 de biți	bun, dar patentat
RC4	Ronald Rivest	1 - 2048 de biți	atentie: anumite chei sunt slabe
RC5	Ronald Rivest	128 - 256 de biți	bun, dar patentat
Rijndael	Daemen and Rijmen	128 - 256 de biți	cea mai bună alegere
Serpent	Anderson, Biham, Knudsen	128 - 256 de biți	foarte puternic
Triplu DES	IBM	168 de biți	a doua alegere
Twofish	Bruce Schneier	128 - 256 de biți	foarte puternic, larg folosit

Fig. 8-16. Câțiva algoritmi criptografici comuni cu cheie simetrică.

8.2.5 Criptanaliza

Înainte de a părăsi subiectul criptografiei cu chei simetrice, merită cel puțin să menționăm patru rezultate recente în criptanaliză. Primul este **criptanaliza diferențială** (Biham și Shamir, 1993). Această tehnică poate fi folosită pentru a ataca orice cifru bloc. Lucrează la început cu o pereche de blocuri de text clar care diferă doar printr-un număr mic de biți și studiază cu grijă ceea ce se întâmplă la fiecare iterație internă pe măsură ce criptarea avansează. În multe cazuri, anumite combinații apar mai des decât altele și această observație conduce la un atac probabilistic.

Al doilea rezultat demn de notat este **criptanaliza liniară** (Matsui, 1994). Aceasta poate sparge DES-ul cu doar 2^{43} texte clare cunoscute. Lucrează prin combinarea XOR a anumitor biți din textul clar și din textul cifrat și examinarea rezultatelor pentru a descoperi tiparele. Când aceasta se face repetat, jumătate din biți trebuie să fie 0 și jumătate să fie 1. Totuși, adeseori, cifrurile introduc o deviație într-o direcție sau în alta și această deviație, cu toate că este mică, poate fi exploatată pentru a reduce efortul. Pentru mai multe detalii a se vedea lucrarea lui Matsui.

Al treilea rezultat este folosirea analizei consumului de energie electrică pentru a afla chei secrete. Calculatoarele folosesc in mod tipic 3 volți pentru a reprezenta un bit 1 și 0 volți pentru a reprezenta un bit 0. De aceea, procesarea unui 1 consumă mai multă energie electrică decât procesarea unui 0. Dacă un algoritm criptografic constă într-o buclă în care biții cheii sunt procesați în ordine, un atacator care înlocuiește ceasul principal de n-GHz cu un ceas lent (de exemplu 100-Hz) și pune mufe crocodil pe sursa procesorului și pinii de masă poate să monitorizeze precis puterea consumată de fiecare instrucțiune mașină. Din aceste date, deducerea cheii este surprinzător de ușoară. Acest tip de criptanaliză poate fi contracarat numai de codificarea atentă a algoritmului în limbaj de asamblare pentru a fi sigur că consumul de putere este independent de cheie și de asemenea independent de toate cheile de rundă individuale.

Al patrulea rezultat este analiza întârzierilor. Algoritmii criptografici sunt plini de instrucțiuni **if** care testează biții cheilor de rundă. Dacă părțile **then** și **else** durează timpi diferiți, prin încetinirea ceasului și măsurarea duratei diverșilor pași se pot deduce cheile de rundă. Odată ce se cunosc toate cheile de rundă, de regulă se poate calcula cheia originală. Analizele de putere și de întârzieri pot fi folosite simultan pentru a face munca mai ușoară. Cu toate că analizele de putere și de întârzieri pot părea exotice, în realitate ele sunt tehnici puternice care pot sparge orice cifru care nu a fost proiectat în mod special pentru a le rezista.

8.3 ALGORITMI CU CHEIE PUBLICĂ

Istoric, distribuția cheilor a fost întotdeauna punctul slab al multor criptosisteme. Indiferent de cât de puternic era un criptosistem, dacă un intrus putea fura cheia, sistemul își pierdea valoarea. Criptologii au considerat întotdeauna ca de la sine înțeles faptul că atât pentru criptare cât și pentru decriptare se folosește aceeași cheie (sau una ușor derivabilă din cealaltă). Dar cheia trebuia distribuită tuturor utilizatorilor sistemului. Astfel, părea a exista întotdeauna următoarea problemă inerentă: cheile trebuia protejate contra furtului dar, în același timp, ele trebuiau distribuite, astfel încât ele nu puteau fi sechestrate într-un seif de bancă.

În 1976, doi cercetători de la Universitatea Stanford, Diffie și Hellman (1976), au propus un tip radical nou de criptosistem în care cheile de criptare și decriptare sunt diferite, iar cheia de decriptare nu poate fi dedusă din cheia de criptare. În propunerea lor, algoritmul (cheia) de criptare, E, și algoritmul (cheia) de decriptare, D, trebuiau să satisfacă trei cerințe. Aceste cerințe pot fi exprimate simplificat după cum urmează:

1. $D(E(P))=P$
2. Este mai mult decât dificil să se deducă D din E.
3. E nu poate fi spart printr-un atac cu text clar ales.

Prima cerință spune că, dacă se aplică D unui mesaj criptat, $E(P)$, se obține textul clar original, P. Fără această proprietate, receptorul legitim nu ar putea decripta textul cifrat. Cea de-a doua cerință vorbește de la sine. Cea de-a treia cerință este necesară deoarece, după cum vom vedea curând, intrușii pot experimenta și testa algoritmul după pofta inimii. În aceste condiții, nu există nici un motiv pentru ca E, cheia de criptare, să nu poată fi făcută publică.

Metoda lucrează astfel: o persoană, să spunem Alice, dorind să primească mesaje secrete, concepe mai întâi doi algoritmi ce satisfac cerințele de mai sus. Algoritmul de criptare și cheia lui Alice sunt făcuți apoi publici, de unde și numele de **criptografie cu cheie publică**. Alice poate de exemplu să își pună cheia publică pe pagina ei personală de pe Web. Vom folosi notația E_A pentru algoritmul de criptare parametrizat de cheia publică a lui Alice. În mod similar, algoritmul (secret) de decriptare parametrizat de cheia privată a lui Alice este D_A. Bob face același lucru, publicând E_B, dar ținând D_B secret.

Să vedem acum dacă putem rezolva problema stabilirii unui canal sigur între Alice și Bob, care nu au mai avut niciodată vreun contact anterior. Atât cheia de criptare a Alicei, E_A, cât și cea a lui Bob, E_B, sunt presupuse a se găsi într-un fișier ce poate fi citit de oricine. Acum Alice ia primul ei mesaj, P, calculează $E_B(P)$ și îl trimite lui Bob. Bob îl decriptează aplicându-i cheia sa secretă D_B [adică, el calculează $D_B(E_B(P))=P$]. Nimeni altcineva nu poate citi mesajul criptat, $E_B(P)$, deoarece sistemul de criptare este presupus puternic și deoarece este prea greu să se deducă $D_B(P)$ din $E_B(P)$ public cunoscut. Pentru a trimite un răspuns R, Bob transmite $E_A(R)$. Alice și Bob pot comunica acum într-o manieră sigură.

În acest punct ar fi poate utilă o observație asupra terminologiei. Criptografia cu cheie publică necesită ca fiecare utilizator să aibă două chei: o cheie publică, folosită de toată lumea pentru a cripta mesajele ce-i sunt trimise, și o cheie secretă, de care utilizatorul are nevoie ca să-și decripteze mesajele. Ne vom referi în mod constant la aceste chei ca fiind cheia *publică* și, respectiv, cheia *privată* și le vom deosebi de cheile *secrete* folosite pentru criptografia convențională cu cheie simetrică.

8.3.1 RSA

Singura problemă este aceea că avem nevoie de algoritmi care să satisfacă complet toate cele trei cerințe. Datorită posibilelor avantaje ale criptografiei cu chei publice, mulți cercetători au lucrat din greu la acest subiect și au fost deja publicați câțiva algoritmi. O metodă bună a fost descoperită de un grup de la MIT (Rivest et. al, 1978). Ea este cunoscută prin inițialele numelor celor trei descoperitori (Rivest, Shamir, Adelman): **RSA**. Metoda a supraviețuit tuturor încercărilor de a o sparge timp de mai mult de un sfert de secol și este considerată foarte puternică. Multe aplicații de securitate se bazează pe ea. Dezavantajul major al acesteia este că necesită chei de cel puțin 1024 de biți pentru o securitate bună (spre deosebire de 128 biți pentru algoritmii cu cheie simetrică), ceea ce o face destul de lentă.

Metoda RSA este bazată pe câteva principii din teoria numerelor. Vom rezuma mai jos modul în care se folosește această metodă; pentru detalii, a se consulta articolul.

1. Se aleg două numere prime, p și q, (de obicei de 1024 biți).
2. Se calculează $n = p \times q$ și $z = (p - 1) \times (q-1)$.
3. Se alege un număr relativ prim cu z și este notat cu d.
4. Se găsește e astfel încât $e \times d = 1 \bmod z$.

Cu acești parametri calculați în avans, suntem gata să începem criptarea. Împărțim textul clar (privit ca șir de biți) în blocuri, astfel încât fiecare mesaj de text clar, P, să intre în intervalul $0 \leq P < n$. Aceasta poate fi făcută grupând textul clar în blocuri de câte k biți, unde k este cel mai mare număr întreg pentru care inegalitatea $2k < n$ este adevărată.

Pentru a cripta mesajul P, se calculează $C = Pe \pmod n$. Pentru a decripta C, se calculează $P = Cd \pmod n$. Se poate demonstra că pentru toți P din intervalul specificat, criptarea și decriptarea sunt funcții inverse una alteia. Pentru a realiza criptarea este nevoie de e și n. Pentru a realiza decriptarea este nevoie de d și n. De aceea, cheia publică constă din perechea (e, n) iar cheia privată din perechea (d, n).

Securitatea metodei este bazată pe dificultatea factorizării numerelor mari. Dacă un criptanalist ar putea factoriza numărul n (public cunoscut), el ar putea găsi p și q, iar din acestea pe z. Cu z și e cunoscuți, criptanalistul îl poate calcula pe d folosind algoritmul lui Euclid. Din fericire, matematicienii au încercat să factorizeze numere mari de cel puțin 300 de ani și experiența acumulată sugerează că aceasta este o problemă mai mult decât dificilă.

După Rivest și colegii săi, factorizarea unui număr de 500 de cifre necesită un timp de calcul de 1025 ani folosind forța brută. În ambele cazuri ei presupun că se folosește cel mai bun algoritm de factorizare și un calculator cu timp de execuție a unei instrucțiuni de 1 μsec. Chiar dacă viteza calculatoarelor va continua să sporească cu un ordin de mărime pe deceniu, vor mai trece secole până când factorizarea unui număr de 500 de cifre va deveni realizabilă, moment în care descendenții noștri vor alege pur și simplu p și q mai mari.

Un exemplu didactic banal pentru algoritmul RSA este dat în fig. 8-17. Pentru acest exemplu am ales $p = 3$ și $q = 11$, rezultând $n = 33$ și $z = 20$. O valoare potrivită pentru d este $d = 7$, deoarece 7 și 20 nu au factori comuni. Cu aceste alegeri, e poate fi găsit prin rezolvarea ecuației $7e = 1 \pmod{20}$, care dă $e = 3$. Textul cifrat, C, pentru textul clar al mesajului, P, este dat de $C = P^3 \pmod{33}$. Textul cifrat este decriptat de către receptor după regula $P = C^7 \pmod{33}$. Fig. prezintă ca exemplu criptarea și decriptarea textului clar „SUZANNE”.

Text clar (P)			Text cifrat (C)		După decriptare	
Simbolic	Numeric	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Simbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
		Calcul efectuat de emițător		Calcul efectuat de receptor		

Fig. 8-17. Un exemplu de algoritm RSA.

Deoarece numerele prime alese pentru acest exemplu sunt prea mici, P trebuie să fie mai mic decât 33, deci fiecare bloc de text clar poate conține doar un singur caracter. Rezultatul este un cifru cu substituție monoalfabetică, nu foarte impresionant. Dacă în locul acestora am fi ales p și $q = 2^{512}$, am fi avut $n = 2^{1024}$, astfel încât fiecare bloc poate fi de până la 1024 de biți sau 128 de caractere de 8 biți, față de 8 caractere pentru DES și 16 caractere pentru AES.

Trebuie subliniat că folosirea RSA în modul descris este similară folosirii unui algoritm simetric în modul ECB - blocuri de intrare identice conduc la blocuri de ieșire identice. De aceea este necesară o anumită formă de înlănțuire pentru criptarea datelor. Totuși, în practică, cele mai multe sisteme bazate pe RSA folosesc criptografia cu cheie publică în principal pentru distribuirea cheilor de sesiune de unică folosință utilizate pentru un algoritm simetric ca AES sau triplu DES. RSA este prea lent pentru a cripta eficient volume mari de date, dar este folosit mult la distribuția de chei.

8.3.2 Alți algoritmi cu cheie publică

Cu toate că RSA este larg răspândit, nu este în nici un caz singurul algoritm cu cheie publică cunoscut. Primul algoritm cu cheie publică a fost algoritmul rucsacului (Merkle și Hellman, 1978). Ideea este că cineva posedă un număr mare de obiecte, fiecare cu greutate diferită. Posesorul codifică mesajul prin selecția secretă a unei submulțimi de obiecte și plasarea lor în rucsac. Greutatea totală a obiectelor din rucsac este făcută publică, ca și lista tuturor obiectelor posibile. Lista obiectelor din rucsac este ținută secretă. Cu câteva restricții suplimentare, problema găsirii unei liste de obiecte cu greutatea dată a fost gândită ca fiind intractabilă computațional și formează baza pentru algoritmul cu cheie publică.

Inventatorul algoritmului, Ralph Merkle, a fost aproape sigur că acest algoritm nu poate fi spart, astfel că el a oferit o recompensă de 100 de dolari oricui îl va putea sparge. Adi Shamir („S”-ul din RSA) l-a spart cu promptitudine și a primit recompensa. Fără a-și pierde curajul, Merkle și-a întărit algoritmul și a oferit o recompensă de 1000 de dolari oricui va putea sparge noul algoritm. Ronald Rivest („R” -ul din RSA) l-a spart cu promptitudine și a luat banii. Merkle nu a îndrăznit să ofere 10000 de dolari pentru următoarea versiune, astfel că „A” (Leonard Adelman) nu a avut noroc. Cu toate acestea, algoritmul rucsacului nu este considerat sigur și este rareori utilizat.

Alte scheme cu cheie publică sunt bazate pe dificultatea calculului logaritmilor discreți (Rabin, 1979). Algoritmii care folosesc acest principiu au fost inventați de El Gamal (1985) și Schnorr (1991).

Există câteva alte scheme, cum ar fi cele bazate pe curbe eliptice (Menezes și Vanstone, 1993), dar cele două categorii majore sunt cele bazate pe dificultatea factorizării numerelor mari și a calculului logaritmilor discreți modulo un număr prim mare. Aceste probleme sunt considerate ca fiind cu adevărat dificile deoarece matematicienii le studiază de mulți ani fără vreun progres notabil.

8.4 SEMNĂTURI DIGITALE

Autenticitatea multor documente legale, financiare și de alt gen este determinată de prezența sau absența unor semnături autorizate scrise de mână. Iar fotocopiile nu sunt valabile. Pentru ca sistemele de mesaje computerizate să înlocuiască transportul fizic al documentelor scrise cu cerneală pe hârtie trebuie găsită o metodă ca documentele să fie semnate într-un mod nefalsificabil.

Problema de a concepe un înlocuitor pentru semnăturile scrise de mână este destul de dificilă. De fapt, este necesar un sistem prin care una din părți poate trimite mesaje „semnate" celeilalte părți astfel încât:

1. Receptorul poate verifica identitatea pe care pretinde a o avea transmițătorul;
2. Transmițătorul nu poate să nege mai târziu că e autorul mesajului;
3. Receptorul nu poate să fi pregătit el însuși mesajul.

Prima cerință este necesară, de exemplu, în sistemele financiare. Atunci când calculatorul unui client ordonă calculatorului unei bănci să cumpere o tonă de aur, calculatorul băncii trebuie să poată să se asigure că acel calculator care dă ordinul aparține într-adevăr companiei al cărei cont va fi debitat. Cu alte cuvinte, banca trebuie să autentifice clientul (și clientul trebuie să autentifice banca).

A doua cerință este necesară pentru a proteja banca împotriva fraudei. Să presupunem că banca cumpără o tonă de aur și imediat după aceea prețul aurului scade brusc. Un client necinstit poate să acuze banca, pretinzând că el nu a emis niciodată vreun ordin de cumpărare de aur. Când banca prezintă mesajul în fața curții, clientul neagă faptul că l-ar fi trimis. Proprietatea că nici o parte a unui contract nu poate nega mai târziu faptul că l-a semnat se numește **nerepudiere** (eng.: nonrepudiation). Schemele cu semnătură digitală pe care le vom studia acum oferă această proprietate.

Cea de-a treia cerință este necesară pentru a proteja clientul în eventualitatea că prețul aurului explodează și banca încearcă să construiască un mesaj în care clientul cere cumpărarea unui lingou de aur în locul unei tone. În acest scenariu de fraudă, banca își păstrează restul de aur pentru ea însăși.

8.4.1 Semnături cu cheie simetrică

Un mod de abordare pentru semnăturile digitale este acela de a avea o autoritate centrală care știe totul și în care oricine are încredere, să spunem Big Brother (*BB*, rom.: fratele cel mare). Fiecare utilizator alege o cheie secretă și o duce personal la biroul *BB*. Astfel, doar Alice și *BB* vor cunoaște cheia secretă a lui Alice, K_A, ș.a.m.d.

Atunci când Alice dorește să trimită un mesaj în clar semnat, *P*, bancherului său, Bob, ea generează $K_A(B, R_A, t, P)$, unde *B* este identitatea lui Bob, R_A este un număr aleator ales de Alice, *t* este o amprentă de timp pentru a asigura prospețimea și $K_A(B, R_A, t, P)$ este mesajul criptat cu cheia ei K_A. Apoi îl trimite, după cum este se arată și în fig. 8-18. *BB* vede că mesajul este de la Alice, îl decriptează și îi trimite lui Bob mesajul ca în figură. Mesajul trimis spre Bob conține textul clar din mesajul lui Alice și, de asemenea, mesajul semnat $K_{BB}(A, t, P)$, unde t este amprenta de timp. Acum Bob rezolvă cererea lui Alice.

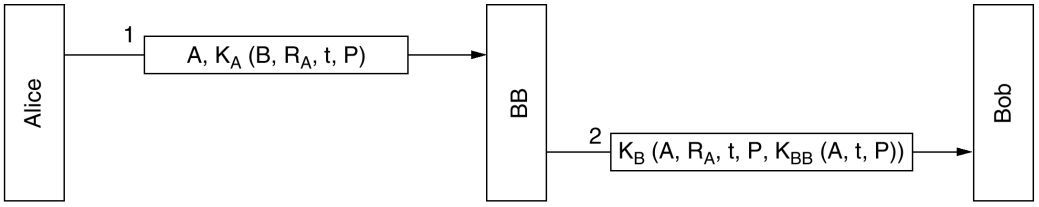


Fig. 8-18. Semnături digitale cu Big Brother.

Ce se întâmplă dacă Alice neagă mai târziu trimiterea mesajului? Primul pas este că toată lumea dă în judecată pe toată lumea (cel puțin în S.U.A.). În final, când cazul ajunge în fața curții și Alice

neagă cu înverșunare că a trimis lui Bob mesajul în dispută, judecătorul îl va întreba pe Bob cum poate fi sigur că mesajul disputat vine de la Alice și nu de la Trudy. Bob va arăta mai întâi că *BB* nu acceptă un mesaj de la Alice decât dacă este criptat cu K_A, așa că nu există nici o posibilitate ca Trudy să-i trimită lui *BB* un mesaj fals ca provenind de la Alice fără ca BB să îl detecteze imediat.

Apoi Bob va aduce în mod incontestabila Probă A, $K_{BB}(A, t, P)$. Bob spune că acesta este un mesaj semnat de *BB* care demonstrează că Alice i-a trimis *P* lui Bob. Judecătorul îi va cere apoi lui *BB* (în care toată lumea are încredere) să decripteze Proba A. Când *BB* va depune mărturie că Bob spune adevărul, judecătorul va de verdictul în favoarea lui Bob. Cazul va fi închis.

O problemă posibilă cu protocolul de semnare din fig. 8-18 apare atunci când Trudy replică oricare din mesaje. Pentru a minimiza această problemă, sunt folosite peste tot amprente de timp. Mai mult, Bob poate verifica toate mesajele recente să vadă dacă R_A a fost folosit în vreunul dintre ele. Dacă da, mesajul respectiv este ignorat deoarece este o replică. De remarcat că Bob va refuza toate mesajele foarte vechi din punct de vedere al amprentei de timp. Pentru a se păzi împotriva atacurilor cu replică instantanee, Bob verifică doar R_A al oricărui mesaj venit, ca să vadă dacă s-a mai primit în ultima oră un astfel de mesaj de la Alice. Dacă nu, Bob poate presupune fără nici un risc că mesajul reprezintă o nouă cerere.

8.4.2 Semnături cu cheie publică

O problemă structurală în folosirea criptografiei cu cheie secretă pentru semnături digitale este aceea că oricine trebuie să se încreadă în Big Brother. Mai mult decât atât, Big Brother poate citi toate mesajele semnate. Cei mai logici candidați pentru a juca rolul lui Big Brother sunt guvernul, băncile, contabilii și avocații. Din păcate, nici una dintre aceste organizații nu inspiră încredere totală tuturor cetățenilor. De aceea ar fi frumos dacă semnarea documentelor nu ar necesita existența unei astfel de autorități de încredere.

Din fericire, criptografia cu cheie publică își poate aduce aici o importantă contribuție. Să presupunem că algoritmii de criptare și decriptare cu cheie publică au proprietatea că $E(D(P))=P$ în plus față de proprietatea uzuală $D(E(P))=P$. (RSA are această proprietate, deci presupunerea nu este nerezonabilă). Presupunând acest lucru, Alice poate trimite un text clar semnat, *P*, lui Bob transmițând $E_B(D_A(P))$. O observație importantă aici este aceea că Alice cunoaște atât propria sa cheie secretă, D_A, cât și cheia publică a lui Bob, E_B, astfel încât construcția acestui mesaj este pentru Alice un lucru realizabil.

Când Bob primește mesajul, el îl transformă, folosindu-și cheia privată, ca de obicei, rezultând $D_A(P)$, după cum este arătat și în fig. 8-19. El memorează acest text într-un loc sigur și apoi aplică E_A pentru a obține textul clar original.

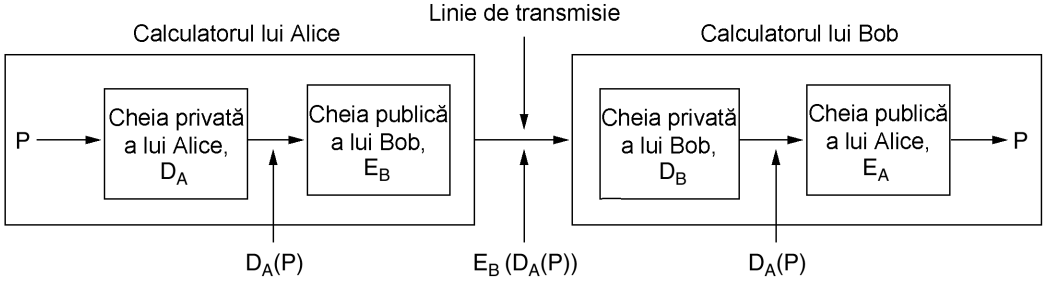


Fig. 8-19. Semnături digitale folosind criptografia cu cheie publică.

Pentru a vedea cum lucrează proprietatea de semnătură, să presupunem că Alice neagă ulterior trimiterea mesajului P lui Bob. Atunci când cazul ajunge în fața curții, Bob poate aduce ca probe atât P cât și $D_A(P)$. Judecătorul poate verifica ușor că Bob are un mesaj valid criptat cu D_A, doar aplicând E_A asupra lui. Deoarece Bob nu știe care este cheia privată a lui Alice, singurul mod prin care Bob poate să fi primit mesajul criptat cu această cheie privată este ca Alice în persoană să-l fi trimis. Cât timp va sta în închisoare pentru minciună și fraudă Alice va avea suficient timp să conceapă noi algoritmi cu cheie publică interesanți.

Cu toate că folosirea criptografiei cu cheie publică pentru semnături digitale este o schemă elegantă, există probleme legate mai degrabă de mediul în care acestea operează decât de algoritmul de la bază. De exemplu, Bob poate dovedi că mesajul a fost trimis de către Alice doar atâta vreme cât DA rămâne secret. Dacă Alice dezvăluie cheia sa secretă acest argument nu va mai fi valabil, deoarece oricine poate să fi trimis mesajul, chiar și Bob.

Problema poate apărea, de exemplu, dacă Bob este agentul de vânzări al lui Alice. Alice îi spune lui Bob să cumpere niște acțiuni. Imediat după aceea, prețul scade vertiginos. Pentru a repudia mesajul său către Bob, Alice face o plângere la poliție, pretinzând că i-a fost spartă casa și furată cheia secretă. În funcție de legislația din țara sau ținutul său, ea poate fi sau nu răspunzătoare legal, în special dacă pretinde că a descoperit spargerea când s-a întors acasă de la muncă, la câteva ore mai târziu.

O altă problemă cu schema de semnătură este ce se întâmplă dacă Alice decide să-și schimbe cheia. A face acest lucru este evident legal și este probabil o idee bună să o facă periodic. Dacă în justiție apare mai târziu un caz, așa cum s-a povestit mai sus, judecătorul va aplica actualul EA la DA(P) și va descoperi că nu se obține P. Bob va fi atunci într-o situație delicată.

În principiu, orice algoritm cu cheie publică poate fi folosit pentru semnături digitale. Standardul de facto în industrie este algoritmul RSA. Multe produse pentru securitate îl folosesc. Totuși, în 1991, NIST (National Institute of Standards and Technology) a propus o variantă a algoritmului cu cheie publică El Gamal pentru noul lor standard **DSS** (**Digital Signature Standard,** rom.: Standard pentru Semnătură Digitală) . El Gamal își bazează securitatea pe dificultatea calculului logaritmilor discreți și nu pe dificultatea factorizării numerelor mari.

Ca de obicei, când guvernul încearcă să impună standarde criptografice, s-a iscat o reacție antagonistă de masă. DSS a fost criticat pentru a fi:

1. Prea secret (NSA a proiectat protocolul pentru folosirea El Gamal).
2. Prea lent (de 10 până la 40 de ori mai lent decât RSA în verificarea semnăturilor).
3. Prea nou (El Gamal nu a fost încă suficient analizat).
4. Prea nesigur (chei fixe de 512 biți).

După o revizuire ulterioară, cel de-al patrulea motiv a fost eliminat fiindcă s-au permis chei de până la 1024 biți. Cu toate acestea, primele două puncte rămân valide.

8.4.3 Rezumate de mesaje

O critică adusă schemelor de semnătură este aceea că adeseori cuplează două funcții distincte: autentificare și confidențialitate. Adesea, autentificarea este necesară, dar confidențialitatea nu. De asemenea, obținerea unei licențe de export este deseori mai ușoară dacă sistemul în chestiune oferă numai autentificare dar nu și confidențialitate. Mai jos vom descrie o schemă de autentificare care nu necesită criptarea întregului mesaj.

Schema este bazată pe ideea unei funcții de dispersie neinversabile care preia o bucată de text clar de lungime arbitrară din care calculează un șir de biți de lungime fixă. Funcția de dispersie, *MD*, adeseori numită **rezumat (digest) al mesajului**, are patru proprietăți importante:

1. Dat fiind *P*, este ușor de calculat *MD*(*P*).
2. Dat fiind *MD*(*P*), este efectiv imposibil de calculat *P*.
3. Dat fiind *P* nimeni nu poate găsi P' astfel încât $MD(P')=MD(P)$.
4. O schimbare la intrare chiar și de 1 bit produce o ieșire foarte diferită.

Pentru a satisface criteriul 3, dispersia trebuie să aibă cel puțin 128 de biți lungime, de preferat chiar mai mult. Pentru a satisface criteriul 4, dispersia trebuie să amestece biții foarte bine, la fel ca algoritmii de criptare cu cheie simetrică pe care i-am văzut.

Calculul rezumatului unui mesaj dintr-o bucată de text clar este mult mai rapidă decât criptarea acelui text clar cu un algoritm cu cheie publică, deci rezumatele de mesaje pot fi folosite pentru a oferi viteză algoritmilor cu semnătură digitală. Pentru a vedea cum lucrează, să considerăm din nou protocolul de semnătură din fig. 8-18. În loc de a semna P cu $K_{BB}(A, t, P)$, *BB* calculează acum rezumatul mesajului aplicând *MD* lui *P*, rezultând *MD*(*P*). *BB* include apoi $K_{BB}(A, t, MD(P))$ ca al cincilea element în lista criptată cu K_B care este trimisă lui Bob, în loc de $K_{BB}(A, t, P)$.

Dacă apare o dispută, Bob poate aduce ca argumente atât *P* cât și $K_{BB}(A, t, MD(P))$. După ce Big Brother l-a decriptat pentru judecător, Bob are *MD*(*P*), care este garantat a fi original și pretinsul *P*. Totuși, deoarece este efectiv imposibil ca Bob să găsească un alt mesaj care să aibă acest rezumat, judecătorul va fi convins ușor că Bob spune adevărul. Folosirea rezumatelor mesajelor în acest mod economisește atât timpul de criptare cât și costurile pentru transport și memorare.

Calculul rezumatelor mesajelor funcționează și în criptosistemele cu chei publice, după cum este arătat și în fig. 8-20. Aici, Alice calculează mai întâi rezumatul de mesaj pentru textul său clar. Apoi ea semnează rezumatul și trimite atât rezumatul semnat cât și textul clar lui Bob. Dacă Trudy îl înlocuiește pe *P* în timpul transferului, Bob va vedea aceasta atunci când va calcula el însuși *MD*(*P*).

Fig. 8-20. Semnături digitale folosind rezumatul mesajului.

MD5

Au fost propuse diverse de funcții pentru calculul rezumatului mesajelor. Cele mai folosite sunt MD5 (Rivest, 1992) și SHA-1 (NIST, 1993). **MD5** este a cincea dintr-o serie de funcții de dispersie proiectate de Ronald Rivest. Operează prin amestecarea biților într-un mod suficient de complicat, astfel încât fiecare bit de ieșire să fie afectat de fiecare bit de intrare. Foarte pe scurt, algoritmul începe prin a umple mesajul până la o lungime de 448 de biți (modulo 512). Apoi lungimea originală a mesajului este adăugată ca un întreg pe 64 de biți, dând o intrare totală a cărei lungime este un multiplu de 512 de biți. Ultimul pas dinaintea începerii calculului este inițializarea unui tampon de 128 de biți la o valoare fixă.

Apoi începe calculul. Fiecare rundă ia un bloc de intrare de 512 de biți și îl amestecă complet cu tamponul de 128 de biți. Ca o măsură suplimentară, este folosită și o tabelă construită folosind func-

ția sinus. Utilizarea unei funcții cunoscute cum este sinus nu se datorează faptului că este mai aleatoare decât un generator de numere aleatoare, ci pentru a evita orice suspiciune că proiectantul a inclus o trapă inteligent ascunsă prin care doar el poate intra. Refuzul IBM-ului de a dezvălui principiile aflate la baza proiectării cutiilor S din DES a dus la destul de multe speculații privind existența trapelor ascunse. Rivest a vrut să evite această suspiciune. Pentru fiecare bloc de intrare sunt efectuate patru runde. Acest proces continuă până când sunt consumate toate blocurile de intrare. Conținutul tamponului de 128 de biți formează rezumatul mesajului.

MD5 este activ deja de peste o decadă și a fost atacat de mulți. S-au găsit câteva vulnerabilități, dar anumiți pași interni previn spargerea lui. Totuși, dacă barierele care rămân în MD5 cad, poate că într-un final va cădea și el. Cu toate acestea, la momentul scrierii acestei cărți el stătea încă în picioare.

SHA-1

Altă funcție majoră pentru calculul rezumatului este **SHA** (**Secure Hash Algorithm,** rom.: Algoritm de Dispersie Sigur), dezvoltată de NSA și acceptată de către NIST ca FIPS 180-1. Ca și MD5, SHA-1 prelucrează datele de intrare în blocuri de câte 512 de biți, dar, spre deosebire de MD5, generează un rezumat de mesaj de 160 de biți. Un mod tipic pentru Alice de a trimite un mesaj nesecret dar semnat lui Bob este ilustrat în fig. 8-21. Aici mesajul ei de text clar este introdus în algoritmul SHA-1 pentru a obține o dispersie SHA-1 de 160 biți. Apoi Alice semnează dispersia cu cheia sa privată RSA și trimite atât mesajul în text clar cât și dispersia semnată lui Bob.

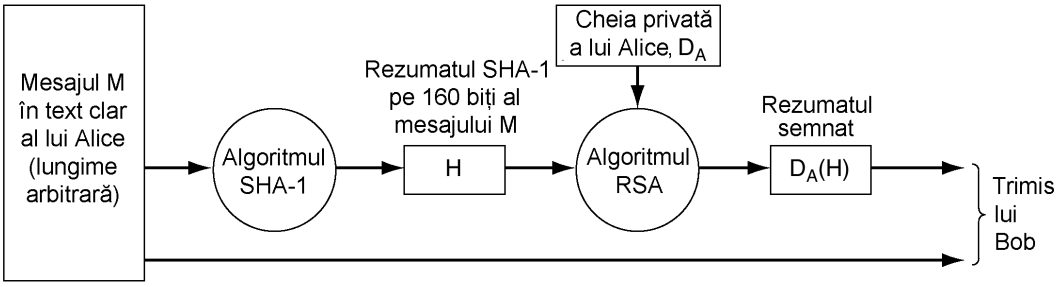


Fig. 8-21. Folosirea lui SHA-1 și RSA pentru a semna mesaje nesecrete

După ce primește mesajul, Bob calculează el însuși dispersia SHA-1 și aplică de asemenea cheia publică a lui Alice asupra dispersiei semnate pentru a obține dispersia originală *H*. Dacă cele două se potrivesc, mesajul este considerat valid. Deoarece nu există nici o cale pentru Trudy de a modifica mesajul (în text clar) în timp ce se trimite și de a produce unul nou care dă ca dispersie H, Bob poate să detecteze ușor orice modificări pe care le-a adus Trudy mesajului. Pentru mesajele a căror integritate este importantă dar al căror conținut nu este secret, se folosește pe larg schema din fig. 8-21. În schimbul unui cost de calcul relativ mic, ea garantează că orice modificări făcute asupra mesajului în text clar în tranzit pot fi detectate cu o probabilitate foarte mare.

Acum să vedem pe scurt cum lucrează SHA-1. Acesta începe prin a completa mesajul, adăugând la sfârșit un bit 1, urmat de atâți biți 0 câți sunt necesari pentru a obține o lungime multiplu de 512 de biți. Apoi se introduce prin OR în cei 64 de biți mai puțin semnificativi un număr de 64 biți conținând lungimea mesajului înaintea completării. În fig. 8-22, este arătat mesajul cu completare la dreapta deoarece textul și cifrele englezești merg de la stânga la dreapta (adică dreapta jos este perceput în general ca sfârșit de cifră). La calculatoare, această orientare corespunde la mașini de tip

big-endian ca SPARC, dar SHA-1 completează întotdeauna sfârșitul mesajului, indiferent de ce tip de endian este mașina.

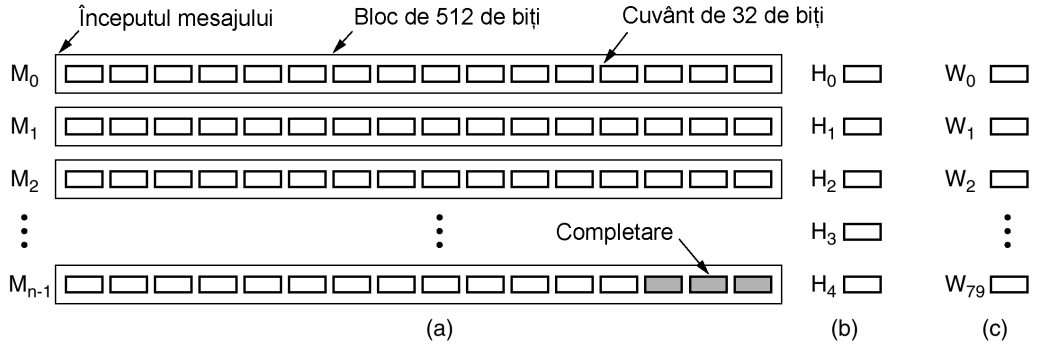


Fig. 8-22. (a) Un mesaj completat până la un multiplu de 512 biți. (b) Variabilele de ieșire. (c) Vectorul de cuvinte

În timpul calculului, SHA-1 păstrează cinci variabile de 32 de biți, H_0 până la H_4, în care se acumulează dispersia. Acestea sunt arătate în fig. 8-22(b). Ele sunt inițializate la constante specificate în standard.

Fiecare din blocurile M_0 până la M_{n-1} este prelucrat pe rând. Pentru blocul curent, primele 16 cuvinte sunt copiate mai întâi la începutul unui vector auxiliar de 80 de cuvinte, W, ca în fig. 8-22(c). Apoi celelalte 64 de cuvinte din W sunt umplute folosind formula

$$W_i = S^i(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

unde $S^b(W)$ reprezintă rotația circulară la stânga, cu *b* biți a cuvântului de 32 de biți *W*. Cinci variabile auxiliare, de la A până la E, sunt apoi inițializate din $H_0 - H_4$ respectiv.

Calculul poate fi prezentat în pseudo-C astfel:

```
for (i = 0; i < 80; i++) {
        temp = S^5(A) + f_i(B, C, D) + E + W_i + K_i;
        E = D; D = C; C = S^30(B); B = A; A = temp;
}
```

unde constantele K_i sunt definite în standard. Funcțiile de amestecare f_i sunt definite astfel:

f_i(B, C, D) = (B AND C) OR (NOT B AND D) ($0 \le i \ge 19$)
f_i(B, C, D) = B XOR C XOR D ($20 \le i \ge 39$)
f_i(B, C, D) = (B AND C) OR (B AND D) OR (C AND D) ($40 \le i \ge 59$)
f_i(B, C, D) = B XOR C XOR D ($60 \le i \ge 79$)

Când toate cele 80 de iterații ale buclei sunt terminate, variabilele A până la E sunt adăugate la H_0 până la H_4, respectiv.

După prelucrarea primului bloc de 512 de biți, se începe următorul. Vectorul W este reinițializat din noul bloc, dar *H* este lăsat cum era. Când acest bloc se termină, se începe următorul, ș.a.m.d., până când toate blocurile de 512 de biți ale mesajului au fost băgate în seamă. Când ultimul bloc a fost terminat, cele cinci cuvinte de 32 de biți din vectorul *H* sunt scoase la ieșire ca dispersia criptografică de 160 biți. Codul complet C pentru SHA-1 este dat în RFC 3174.

Noi versiuni ale SHA-1 sunt în curs de alcătuire pentru dispersii de respectiv 256, 384 și 512 biți.

8.4.4 Atacul zilei de naștere

În lumea criptografiei, nimic nu este ceea ce pare a fi. S-ar putea crede că sunt necesare 2m operații pentru a falsifica un rezumat de m biți. De fapt, 2m/2 sunt suficiente dacă se folosește atacul zilei de naștere, o abordare publicată de Yuval (1979) în lucrarea sa „How to Swindle Rabin" (rom.: Cum să-l jefuiești pe Rabin).

Ideea acestui atac vine de la o tehnică pe care profesorii de matematică o folosesc adeseori în cursurile lor de teoria probabilităților. Întrebarea este: Câți studenți trebuie să fie într-o clasă pentru ca probabilitatea de a exista doi studenți cu aceeași dată de naștere să fie 1/2? Mulți studenți se așteaptă ca răspunsul să fie ceva peste 100. De fapt, teoria probabilităților spune că trebuie să fie exact 23. Fără a face o analiză riguroasă, intuitiv, cu 23 de oameni se pot forma $(23\times23)/2 = 253$ perechi diferite, fiecare dintre ele având o probabilitate de 1/365 să fie cea potrivită. În această lumină, rezultatul nu mai este deloc surprinzător.

Mai general, dacă există o corespondență între intrări și ieșiri cu n intrări (oameni, mesaje etc.) și k ieșiri (zile de naștere, rezumate etc.), există n(n-2)/2 perechi de intrare. Dacă n(n-2)/2 > k, șansa de a avea cel puțin o potrivire este destul de mare. Astfel, cu aproximație, o potrivire este posibilă pentru $n > \sqrt{k}$. Acest rezultat înseamnă că un rezumat de 64 de biți poate fi probabil spart prin generarea a aproape 232 mesaje și căutând două cu același rezumat.

Să vedem acum un exemplu practic. Departamentul de Știința Calculatoarelor de la Universitatea de Stat are disponibil un post de profesor titular și există doi candidați la el, Tom și Dick. Tom a fost angajat cu doi ani înaintea lui Dick, așa că el merge primul la verificare. Dacă reușește, Dick are ghinion. Tom știe că șefa departamentului, Marylin, are o impresie bună despre activitatea sa, așa că o roagă să îi scrie o scrisoare de recomandare pentru decan, cel care va decide în cazul său. Odată trimise, toate scrisorile devin confidențiale.

Marylin spune secretarei, Ellen, să scrie Decanului o scrisoare, subliniind ceea ce dorește. Când este gata, Marylin o va revizui, va calcula și va semna un rezumat de 64 de biți și îl va trimite decanului. Ellen poate trimite scrisoarea mai târziu, prin poștă electronică.

Din nefericire pentru Tom, Ellen are o idilă cu Dick și i-ar plăcea să îl facă pe Tom să eșueze, așa că ea scrie o scrisoare cu următoarele 32 de opțiuni cuprinse între paranteze drepte.

Dear Dean Smith,

This [*letter* | *message*] is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* | *worked with*] Prof. Wilson for [*about* | *almost*] six years. He is an [*outstanding* | *excellent*] researcher of great [*talent* | *ability*] known [*worldwide* | *internationally*] for his [*brilliant* | *creative*] insights into [*many* | *a wide variety of*] [*difficult* | *challenging*] problems.

He is also a [*highly* | *greatly*] [*respected* | *admired*] [*teacher* | *educator*]. His students give his [*classes* | *courses*] [*rave* | *spectacular*] reviews. He is [*our* | *the Department's*] [*most popular* | *best-loved*] [*teacher* | *instructor*].

[*In addition* | *Additionally*] Prof. Wilson is a [*gifted* | *effective*] fund raiser. His [*grants* | *contracts*] have brought a [*large* | *substantial*] amount of money into [*the* | *our*] Department. [*This money has* | *These funds have*] [*enabled* | *permitted*] us to [*pursue* | *carry out*] many [*special* | *important*] programs, [*such as* | *for example*] your State 2000 program. Without these funds we would [*be enable* | *not be able*] to continue this program which is so [*important* | *essential*] to both of us. I strongly urge you to grant him tenure.

Din nefericire pentru Tom, imediat după ce Ellen termină de alcătuit și introdus mesajul, ea scrie de asemenea un al doilea mesaj:

Dear Dean Smith,

This [*letter* | *message*] is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* | *worked with*]. Prof. Wilson for [*about* | *almost*] six years. He is an [*poor* | *weak*] researcher not well known in his [*field* | *area*]. His research [*hardly ever* | *rarely*] shows [*insight in* | *understanding of*] the [*key* | *major*] problems of [*the* | *our*] day.

Furthermore, he is not a [*respected* | *admired*] [*teacher* | *educator*]. His students give his [*classes* | *courses*] [*poor* | *bad*] reviews. He is [*our* | *the Department's*] least popular [*teacher* | *instructor*], known [*mostly* | *primarily*] within [*our* | *the*] Department for his [*tendency* | *propensity*] to [*ridicule* | *embarrass*] students [*foolish* | *imprudent*] enough to ask questions in his classes.

[In addition | *Additionally*] Tom is a [*poor* | *marginal*] fund raiser. His [*grants* | *contracts*] have brought only a [*meager* | *insignificant*] amount of money into [*the* | *our*] Department. Unless new [*money is* | *funds are*] quickly located, we must have to cancel essential programs such as your State 2000 program. Unfortunately, under these [*conditions* | *circumstances*]. I cannot in good [*conscience* | *faith*] recommend him to you for [*tenure* | *a permanent position*].

Ellen pune calculatorul să calculeze în timpul nopții 2^{32} rezumate de mesaj pentru fiecare scrisoare. Există șanse ca un rezumat al primei scrisori să se potrivească cu un rezumat al celei de-a doua scrisori. Dacă nu, ea poate să adauge mai multe opțiuni și să încerce din nou în timpul weekendului. Să presupunem că găsește o potrivire. Să notăm scrisoarea „bună" cu *A* și scrisoarea „rea" cu *B*.

Ellen trimite acum scrisoarea *A* lui Marilyn pentru aprobare. Scrisoarea B o păstrează complet secretă, nearătând-o nimănui. Desigur că Marilyn o aprobă, calculează rezumatul de 64 de biți, semnează rezumatul și trimite rezumatul semnat decanului Smith. Independent, Ellen îi trimite decanului Smith scrisoarea *B (*și nu scrisoarea *A*, cum trebuia).

După ce primește scrisoarea și rezumatul semnat, decanul rulează algoritmul de calcul al rezumatului pentru scrisoarea *B*, vede că se potrivește cu ceea ce i-a trimis Marilyn și îl dă afară pe Tom. Decanul nu înțelege că Ellen a reușit să genereze două scrisori cu același rezumat și i-a trimis o scrisoare diferită decât cea pe care a văzut-o și aprobat-o Marilyn. (Sfârșit opțional: Ellen îi povestește lui Dick ceea ce a făcut. Dick este revoltat și rupe relația cu ea. Ellen este furioasă și se confesează lui Marilyn. Marilyn îl sună pe Decan. Tom obține până la urmă postul). Atacul zilei de naștere este nefezabil la MD5 deoarece, chiar și la un miliard de rezumate pe secundă, ar trebui peste 500 de ani pentru a calcula 2^{64} rezumate pentru cele două scrisori, cu 64 de variante fiecare, și nici atunci succesul nefiind garantat. SHA-1 este mai bun (deoarece este mai lung).

8.5 GESTIONAREA CHEILOR PUBLICE

Criptografia cu chei publice face posibilă comunicația sigură între persoane care nu împart o cheie comună. De asemenea face posibilă semnarea mesajelor fără prezența unei a treia părți de încredere. În fine, rezumatul mesajului semnat face posibilă verificarea ușoară a integrității mesajelor primite.

Oricum, există o problemă peste care am trecut un pic cam repede: dacă Alice și Bob nu se cunosc unul pe altul, cum își pot afla cheile publice pentru a porni procesul de comunicație? Soluția evidentă – puneți cheia publică pe web – nu merge datorită următorului motiv. Presupunem că Alice vrea să caute cheia publică a lui Bob pe site-ul lui de web. Cum face ea acest lucru? Începe prin a tasta adresa de web a lui Bob. Browserul ei va căuta adresa DNS a paginii principale a lui Bob și va trimite o cerere de tip *GET*, așa cum se indică în fig. 8-23. Din păcate, Trudy interceptează cererea și răspunde cu o falsă pagină principală, probabil o copie a paginii principale a lui Bob în care s-a înlocuit cheia publică a lui Bob cu cheia publică a lui Trudy. Când Alice va cripta primul ei mesaj cu E_T, Trudy îl va decripta, îl va citi, îl va recripta cu cheia publică a lui Bob și îl va trimite apoi lui Bob, care nu este capabil să-și dea seama că Trudy i-a citit mesajele primite. Și mai grav, Trudy poate modifica mesajele înainte de a le recripta pentru Bob. În mod evident, este nevoie de un mecanism care să asigure un schimb sigur de chei publice.

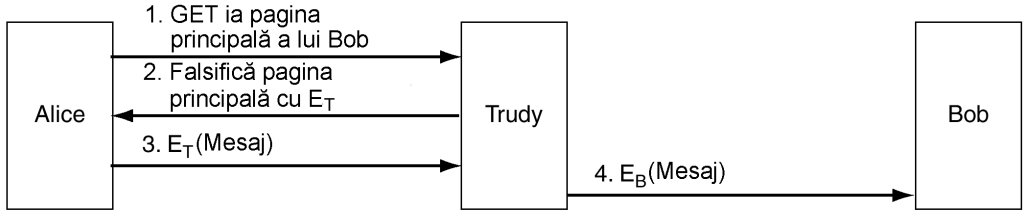


Fig. 8-23. O modalitate prin care Trudy sparge infrastructura cu chei publice

8.5.1 Certificate

Ca o primă încercare de distribuire sigură a cheilor publice, ne putem imagina un centru de distribuție a cheilor disponibil 24 de ore pe zi pentru a oferi chei publice la cerere. Una dintre multele probleme ale acestei soluții este lipsa de scalabilitate și faptul că centrul de distribuție de chei va deveni rapid un punct de gâtuire a procesului de distribuție de chei. De asemenea, dacă centrul se va opri vreodată, securitatea Internetului se va pierde in mod subit.

Din cauza acestor motive, s-a dezvoltat o soluție diferită, care nu presupune un centru de distribuție de chei care să fie disponibil tot timpul. De fapt, acesta nu trebuie să fie disponibil deloc. În schimb, soluția aleasă certifică faptul că o cheie publică aparține unei anume persoane, companii sau altei organizații. O organizație care certifică chei publice este numită **CA (Certification Authority, rom.: Autoritate de Certificare)**.

Subsemnatul certific faptul că cheia publică
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
aparține lui
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

Rezumatul SHA-1 al certificatului de mai sus semnat cu cheia privată a CA

Fig. 8-24. Un certificat posibil și rezumatul său semnat

De exemplu, să presupunem că Bob vrea să-i permită lui Alice și altor persoane să comunice în mod sigur cu el. El poate să meargă la un CA cu cheia sa publică însoțită de pașaport sau permisul de conducere și să ceară să fie certificat. CA-ul va emite un certificat similar cu cel din fig. 8-24 și va semna rezumatul de tip SHA-1 al certificatului cu cheia sa secretă. Bob va plăti prețul cerut de CA și va obține o dischetă conținând certificatul și rezumatul semnat.Scopul principal al unui certificat este să facă legătura între o cheie publică și o entitate (individ, companie etc.). Certificatele în sine nu sunt secrete sau protejate. Bob poate, de exemplu, să decidă să-și pună noul certificat pe sit-ul său web, cu o legătură către pagina sa principală în care sa afirme: "Apasă aici pentru certificatul chei mele publice". Ca rezultat se vor returna atât certificatul cât și semnătura sa (codul de dispersie semnat SHA-1 al certificatului).

Să mai analizam încă o dată scenariul din fig. 8-23. Când Trudy interceptează cererea lui Alice pentru pagina principală a lui Bob, ce poate face ea acum? Își poate pune propriul ei certificat și semnătura pe pagina falsă, dar când Alice va citi certificatul își va da seama imediat că nu vorbește cu Bob deoarece numele lui Bob nu se găsește în certificat. Trudy poate modifica pagina lui Bob din mers, înlocuind cheia publică a lui Bob cu propria ei cheie. Oricum, când Alice va rula algoritmul SHA-1 pe certificat, va obține un rezumat care va fi diferit de cel obținut prin aplicarea cheii publice a CA-ului asupra semnăturii certificatului. Cum Trudy nu deține cheia privată a CA-ului, ea nu are nici o modalitate de a genera un bloc de semnătură care să conțină codul de dispersie modificat de ea, prin înlocuirea propriei sale chei publice în pagina de web. În acest fel Alice poate fi sigură ca are cheia publică a lui Bob și nu pe cea a lui Trudy sau a altcuiva. Cum am promis, această schemă nu presupune un CA disponibil mereu pentru verificare, eliminându-se astfel o posibilă gâtuire.

În timp ce funcția standard a unui certificat este de a face legătura între o cheie publică și o entitate, un certificat poate face legătura între o cheie publică și un atribut. De exemplu un certificat poate afirma: "Această cheie publică aparține cuiva care are vârsta de peste 18 ani. Ea poate fi folosită pentru a dovedi că posesorul cheii private nu este minor și că are acces la materiale interzise copiilor și așa mai departe" , fără a dezvălui identitatea posesorului certificatului. În mod obișnuit, persoana care deține un certificat, îl va trimite unui sit Web, persoană sau proces de administrare, în care se ține cont de vârstă. Acel sit (persoana sau procesul de administrare) va genera un număr aleator și îl va cripta cu cheia publică prezentă în certificat. Dacă posesorul certificatului este capabil să decripteze mesajul și să-l trimită înapoi, acest lucru va fi o dovada că posesorul deține într-adevăr atributul din certificat. Ca alternativă, numărul aleator poate fi folosit pentru a genera o cheie de sesiune pentru a garanta comunicația.

Un alt exemplu în care certificatul poate conține un atribut este cazul sistemelor distribuite orientate pe obiecte. Fiecare obiect are în mod normal mai multe metode. Proprietarul unui obiect poate oferi fiecărui client un certificat care să conțină o hartă de biți cu metode ce sunt permise clientului respectiv și poate face legătura între cheia publică și harta de biți prin folosirea unui certificat semnat. Și în acest caz, dacă posesorul certificatului poate dovedi că este în posesia cheii secrete corespunzătoare, el va avea dreptul să execute metodele identificate de harta de biți. Certificatul are proprietatea că identitatea posesorului nu trebuie să fie cunoscută, proprietate utilă în situații în care confidențialitatea este importantă.

8.5.2 X.509

Dacă fiecare persoană care dorește ceva semnat ar merge la CA pentru diverse tipuri de certificate, gestionarea tuturor tipurilor de formate ar deveni curând o problemă. Pentru a rezolva această

problemă, s-a proiectat și aprobat de către ITU un standard pentru certificate. Standardul se numește X.509 și este folosit pe scara largă în Internet. De la prima standardizare din 1998, au existat trei versiuni. Vom discuta mai departe despre V3.

X.509 a fost foarte mult influențat de lumea OSI, împrumutând unele din cele mai proaste trăsături (ex. politica de nume și codificarea). În mod surprinzător, IETF a fost de acord cu X.509, chiar dacă în alte domenii, de la adresele mașinilor la protocoalele de transport și formatul poștei electronice, IETF ignoră OSI și încearcă să facă lucrurile corect. Versiunea IETF pentru X.509 este descrisă în RFC 3280.

În principal, X.509 este o modalitate de a descrie certificate. Câmpurile principale dintr-un certificat sunt listate in fig. 8-25. Descrierea dată aici ar trebui să ofere o idee generală a ceea ce fac câmpurile respective. Pentru informații adiționale, vă rog să consultați standardul în sine sau RFC 2459.

Câmp	Semnificație
Versiune	Ce versiune de X.509 este utilizată
Număr Serial	Acest număr împreună numele CA-ului identifică în mod unic certificatul
Algoritm de semnare	Algoritmul folosit la semnarea certificatului
Emitent	Numele X.500 al CA-ului
Perioada de validitate	Momentele de început și sfârșit ale perioadei de validitate
Numele subiectului	Entitatea care este certificată
Cheia publică	Cheia publică a subiectului și ID-ul algoritmului folosit
ID emitent	Un identificator opțional identificând în mod unic emitentul certificatului
ID subiect	Un identificator opțional identificând în mod unic subiectul certificatului
Extinderi	Au fost definite mai multe extinderi
Semnătura	Semnătura certificatului (semnat cu cheia privată a CA-ului)

Fig. 8-25. Câmpurile principale dintr-un certificat X.509

De exemplu, daca Bob lucrează în departamentul de împrumuturi al Băncii Bani, adresa sa X.500 poate sa fie:

/c=US/O=MoneyBanc/OU=Loan/CN=Bob/

unde C indică țara, O indica organizația, OU reprezintă o unitate din organizație, și CN este folosit drept numele comun (eng.: common name). CA-urile și alte entități sunt denumite similar. O problemă mare cu numele X.500 este că, dacă Alice vrea să-l contacteze pe bob@moneybank.com și are un certificat conținând un nume X.500, nu este evident că acel certificat se referă la acel Bob pe care vrea ea să-l contacteze. Din fericire, începând cu versiunea 3, sunt permise numele DNS în loc de numele X.500, deci această problemă ar putea să dispară.

Certificatele sunt codificate folosind **OSI ASN.1 (eng.: Abstract Syntax Notation 1, rom.: Notația sintactică abstractă 1)**, care poate fi văzută ca o structură C, cu excepția unei notații foarte specifice și detaliate. Mai multe informații despre X.509 pot fi găsite in (Ford și Baum, 2000).

8.5.3 Infrastructuri cu chei publice

Existența unei singure Autorități de certificare care să emită toate certificatele din lume nu este evident o bună soluție. Ea ar ceda datorită încărcării mari și va fi în același timp și un punct central de defectare. O soluție posibilă este existența mai multor CA-uri, toate rulând în cadrul aceleași organizații și folosind aceeași cheie privată pentru semnarea certificatelor. Cu toate că se rezolvă problema încărcării și a defectării, această soluție introduce o problemă nouă: dezvăluirea cheilor. Dacă ar exista o zeci de servere împrăștiate prin lume, toate deținând cheia privată a CA-ului, posibilitatea

furtului cheii private sau a dezvăluirii ei va crește foarte mult. Cum compromiterea acestei chei va ruina securitatea infrastructurii electronice mondiale, existența unei singure CA centrale este un risc foarte mare.

În plus, ce organizații vor juca rolul de CA? Este foarte greu de imaginat că orice autoritate este acceptată la nivel mondial ca legitimă și de încredere. În unele țări oamenii vor insista să fie guvernamentală, în timp ce în altele vor insista să nu fie o organizație guvernamentală.

Din cauza acestor motive, a fost dezvoltată o altă variantă de certificare a cheilor publice. Denumirea generală este de **PKI** (eng.: **Public Key Infrastructure**, rom.: **Infrastructură cu chei publice**) . În această secțiune vom prezenta pe scurt cum funcționează PKI în general, deși au existat mai multe propuneri care vor face ca detaliile să evolueze în timp.

O PKI are mai multe componente, incluzând utilizatorii, CA-urile, certificatele și directoarele. Scopul unei PKI este să ofere o structurare a acestor componente și să definească standarde pentru diferite documente și protocoale. O formă particulară de PKI este o ierarhie de CA-uri, așa cum se arată în fig. 8-26. În acest exemplu se prezintă trei niveluri, dar în practică pot fi mai multe sau mai puține. CA-ul din vârf, rădăcina, certifică a doua CA, pe care o vom denumi **RA** (eng.: **Regional Authority**, rom.: **Autoritate Regională**) deoarece va răspunde de anumite regiuni geografice, precum o țară sau un continent. Acest termen nu este standard; de fapt nici un termen referitor la diferite nivele din arbore nu este standardizat. Aceste RA-uri certifică de fapt adevăratele CA-uri, care vor emite certificate X.509 pentru organizații și indivizi. Când rădăcina autorizează o nouă RA, ea va genera un nou certificat X.509 care atestă că a aprobat RA-ul, incluzând în el noua cheie publică a RA-ului, îl va semna și îl va trimite RA-ului. Similar, RA-ul aprobă noi CA-uri, generează și semnează certificate care atestă aprobarea și conțin cheia publică a CA-ului.

PKI-ul nostru funcționează în modul următor. Să presupunem că Alice are nevoie de cheia publică a lui Bob pentru a comunica cu acesta, deci ea va căuta și va găsi un certificat semnat de CA 5 care să conțină cheia publică respectivă. Dar Alice nu a auzit niciodată de CA 5. Din câte știe ea, CA 5 poate fi și fiica de 10 ani a lui Bob. S-ar putea duce la CA5 și să-i pretindă să își dovedească legitimitatea. CA 5 răspunde cu certificatul obținut de la RA 2, care conține cheia publică a lui CA 5. Acum, deținând cheia publică a lui CA 5, ea poate verifica dacă certificatul lui Bob este într-adevăr semnat de CA 5 și datorită acestui fapt este legal.

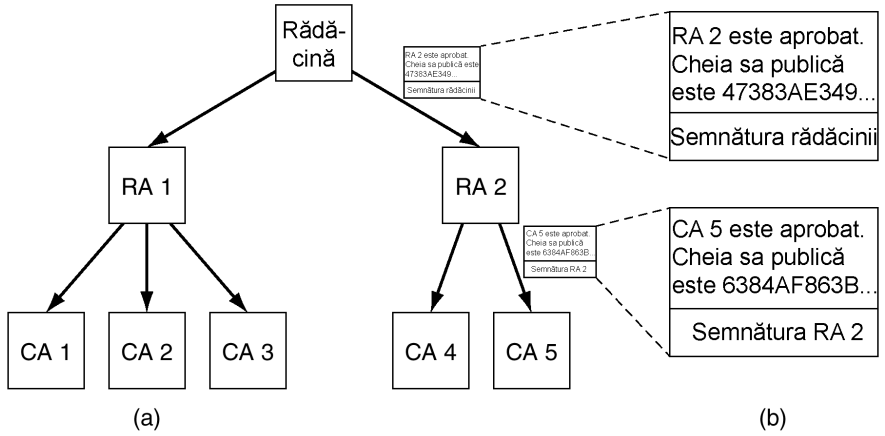


Fig. 8-26. (a) O PKI ierarhică (b) Un lanț de certificate

Dar dacă RA 2 este copilul de 12 ani al lui Bob? Deci pasul următor pentru ea este de a cere ca RA 2 să-și dovedească legitimitatea. Răspunsul la întrebarea ei este certificatul semnat de rădăcină și care conține cheia publică a lui RA 2. Acum Alice este sigură ca ea deține cheia publică a lui Bob.

Dar cum știe Alice sa găsească cheia publică a rădăcinii? Magie. Se presupune că toată lumea cunoaște cheia publică a rădăcinii. De exemplu, programul său de navigare ar fi putut fi vândut cu cheia publică a rădăcinii înglobată în el.

Bob este un tip prietenos și nu vrea să-i provoace lui Alice multă bătaie de cap. El știe că ea trebuie să verifice CA 5 și RA 2, deci pentru a-i economisi timp, el colectează certificatele necesare și îi transmite încă două certificate alături de al său. Acum ea își poate folosi propria ei informație cu privire la cheia publică a rădăcinii pentru a verifica certificatul din vârful ierarhiei și cheia publică conținută în acesta pentru a verifica al doilea certificat. În acest fel, Alice nu mai trebuie să contacteze pe nimeni pentru a efectua verificarea. Pentru că toate certificatele sunt semnate, ea poate detecta cu ușurința orice încercare de falsificare a conținutului acestora. Un lanț de certificate care duce la rădăcină, ca în acest exemplu, este denumit câteodată **lanț de încredere** sau **cale de certificare.** Tehnica este larg răspândită în practică.

Desigur, mai rămâne problema cine va juca rolul de rădăcină. Soluția este să nu avem o singură rădăcină, ci să avem mai multe, fiecare cu propriile sale RA-uri și CA-uri. De fapt, programele de navigare moderne vin preîncărcate cu cheile publice a peste 100 de rădăcini, denumite câteodată **puncte de încredere**. În acest fel, poate fi evitată existența la nivel mondial a unei singure autorități de încredere.

Dar apare acum problema cum poate decide producătorul programului de navigare care dintre punctele sugerate sunt de încredere și care nu. Totul se reduce la încrederea utilizatorului în faptul că producătorul programului de navigare este capabil să facă alegeri înțelepte și nu doar să aprobe toate punctele de încredere care au plătit o taxă pentru includere. Majoritatea programelor de navigare permit utilizatorilor să inspecteze cheile rădăcinilor (în mod obișnuit sub formă de certificate semnate de rădăcină) și să le șteargă pe cele care par dubioase.

Directoare

O altă problemă pentru orice PKI este locul în care se stochează certificatele (și lanțurile care duc către un punct de încredere cunoscut). O posibilitate este ca fiecare utilizator să-și stocheze propriile certificate. Cu toate că acest lucru este sigur (de ex. nu există nici o posibilitate ca un utilizator să falsifice certificatele semnate fără a fi detectat), nu este prea convenabil. O alternativă care a fost propusă este folosirea DNS ca director pentru certificate. Înainte de a-l contacta pe Bob, Alice probabil îi va căuta adresa IP folosind DNS, deci de ce să nu întoarcă DNS-ul întregul lanț de certificate odată cu adresa IP?

Unele persoane consideră că aceasta este soluția corectă, dar alții preferă servere de directoare specializate a căror îndatorire este doar gestionarea de certificate X.509. Directoarele de acest fel pot oferi servicii de căutare bazate pe proprietățile numelor X.500. De exemplu, în teorie un astfel de director poate răspunde la o întrebare de tipul: "Dă-mi o listă a persoanelor numite Alice care lucrează în departamentul de vânzări oriunde în Canada sau S.U.A.". LDAP poate fi un candidat care să stocheze acest tip de informații.

Revocarea

Lumea reală este plină de certificate, la fel ca în cazul pașapoartelor și al carnetelor de conducere. Câteodată aceste certificate pot fi revocate, de exemplu, carnetul de conducere poate fi suspendat pentru conducere sub influența alcoolului sau pentru o altă greșeală de conducere. Aceeași problemă apare și în lumea digitală: emitentul unui certificat poate decide dacă să-l revoce

problemă apare și în lumea digitală: emitentul unui certificat poate decide dacă să-l revoce în cazul în care persoana sau organizația care îl deține a abuzat de el într-un anume mod. El poate fi de asemenea revocat dacă cheia privată a subiectului a fost compromisă (demascată), sau mai rău, dacă cheia privată a CA-ului a fost compromisă. Atunci, o PKI trebuie să fie capabilă să trateze problema revocării.

Primul pas în această direcție este ca fiecare CA să emită periodic o **CRL** (eng.: **Certificate Revocation List,** rom.: **Listă de Certificate Revocate**) conținând numerele seriale ale tuturor certificatelor revocate de aceasta. Cum certificatele conțin data de expirare, CRL-ul trebuie să conțină doar numerele seriale ale certificatelor care încă nu au expirat. Odată ce perioada de validitate a expirat, un certificat este automat invalidat, deci nu mai este necesară nici o distincție între acelea care tocmai au expirat și cele are au fost de fapt revocate. În ambele cazuri, certificatele nu mai pot fi folosite în continuare.

Din păcate, introducerea CRL-ului înseamnă că un utilizator care dorește să folosească un certificat trebuie acum să consulte CRL-ul pentru a vedea dacă certificatul a fost revocat. Dacă a fost revocat, nu mai trebuie folosit. Oricum, chiar dacă certificatul nu este în listă, el poate să fi fost revocat chiar după ce lista a fost publicată. Atunci, singura modalitate rămâne întrebarea către CA. La o nouă folosire a aceluiași certificat, CA-ul trebuie întrebat din nou, pentru este posibil ca certificatul să fi fost revocat cu câteva secunde în urmă.

O altă complicație este că un certificat revocat poate în principiu să fie restabilit, de exemplu, dacă a fost revocat datorită neplății unei taxe care acum s-a plătit. Având de a face cu revocarea (și posibil cu restabilirea) se elimină una dintre cele mai bune proprietăți ale certificatelor, adică faptul că ele pot fi folosite fără a fi necesar sa se contacteze o RA.

Unde trebuie stocate CRL-urile? Un loc bun ar putea fi acela în care se stochează certificatele însele. O strategie este ca CA-ul să publice periodic CRL-uri și să permită directoarelor să înlăture certificatele revocate. Dacă nu sunt folosite directoare pentru stocarea de certificate, CRL-urile pot fi ținute în diverse locuri convenabile din rețea. Cum o CRL este ea însăși un document, dacă este falsificată, falsul se poate detecta ușor.

Dacă certificatele au durate de viață lungi, și CRL-urile vor avea perioade lungi. De exemplu, când cărțile de credit sunt valide pentru 5 ani, numărul de nerezolvări ale revocărilor va fi mult mai mare decât dacă se emit cărți de credit noi la fiecare 3 luni. Un mod standard de a aborda CRL-urile lungi este de a emite rar o listă principală, dar de a-i face modificări mult mai des. Acest lucru reduce lățimea de bandă necesară distribuirii CRL-urilor.

8.6 SECURITATEA COMUNICAȚIEI

Am terminat acum studiul asupra instrumentelor comerciale. Cele mai importante tehnologii și protocoale au fost prezentate. Restul capitolului se referă la modul de aplicare în practică a acestor tehnologii pentru a se asigura securitatea rețelei, plus câteva gânduri despre aspectele sociale ale securității prezentate la sfârșitul capitolului.

În următoarele patru secțiuni, ne vom axa pe securitatea comunicației, deci cum să transferăm biții confidențial și fără a fi modificați, de la sursă la destinație și cum să ținem biții nedoriți în fața

uși. Fără discuție că acestea nu sunt singurele idei despre securitatea în rețele, dar cu siguranță sunt printre cele mai importante, constituind un bun loc de plecare.

8.6.1 IPsec

IETF a știut de ani de zile că securitatea lipsea din Internet. Adăugarea ei nu a fost ușoară deoarece a izbucnit un război cu privire la locul de plasare al acesteia. Majoritatea experților în securitate credeau că pentru a fi într-adevăr sigure, criptarea și verificarea integrității trebuie să fie capăt la capăt (de ex. nivelul aplicație). Adică, procesul sursă criptează și/sau protejează datele din punct de vedere al integrității și le trimite către procesul de destinație, unde sunt decriptate și/sau verificate. Orice falsificare între aceste două procese, incluzând sistemele lor de operare, poate fi detectată. Problema cu această abordare este că presupune schimbarea tuturor aplicațiilor pentru a le securiza. În această idee, următoarea abordare posibilă este plasarea criptării pe nivelul transport sau într-un nou nivel între nivelul aplicație și nivelul transport, păstrând mecanismul tot capăt la capăt, dar fără a mai fi necesare schimbări ale aplicațiilor.

Ideea opusă este că utilizatorii nu înțeleg securitatea și că nu vor fi capabili să o folosească în mod corect și nimeni nu dorește să modifice programele existente în nici un fel, deci nivelul rețea trebuie să autentifice și/sau cripteze pachetele fără nici o implicare din partea utilizatorilor. După ani de controverse, această idee a câștigat suficient suport astfel încât a fost definit un standard de securitate pentru nivelul rețea. În parte, argumentul a fost că având criptarea la nivelul rețea nu împiedică utilizatorii conștienți de problema securității să o folosească corect și îi ajută într-un anumit grad pe utilizatorii neavizați.

Rezultatul acestui război a fost un proiect numit IPsec (IP securizat), care este descris în RFC-urile 2401, 2402 și 2406, printre altele. Nu toți utilizatorii doresc criptarea (pentru că este costisitoare din punct de vedere al puterii de calcul). În loc să fie opțională, s-a decis să se impună criptarea permanent, dar cu posibilitatea de folosire unui algoritm nul. Algoritmul nul este descris și lăudat pentru simplitatea sa, ușurința de implementare și viteza mare în RFC 2410.

Proiectul complet IPsec este o cadru de lucru pentru mai multe servicii, algoritmi și granularități. Motivul pentru servicii multiple este că nu toată lumea dorește să plătească prețul necesar tuturor serviciilor, deci serviciile sunt disponibile la cerere. Serviciile principale sunt confidențialitatea, integritatea datelor, și protejarea lor de atacul prin replicare (un intrus poate replica un dialog). Toate acestea se bazează pe criptografia cu chei simetrice pentru că înalta performanță este un lucru crucial.

Motivul pentru care există mai mulți algoritmi este că un algoritm care acum pare sigur poate fi spart in viitor. Făcând IPsec-ul independent de algoritmi, cadrul de lucru poate supraviețui chiar dacă mai târziu un algoritm este spart.

Motivul pentru care există mai multe granularități se datorează posibilității de a se proteja o singură conexiune TCP, tot traficul dintre două gazde, sau tot traficul dintre o pereche de rutere securizate, printre alte posibilități.

Un aspect destul de surprinzător al IPsec-ului este că deși se găsește pe nivel IP, el este orientat pe conexiune. De fapt, nu este chiar așa de surprinzător deoarece pentru a exista securitate, o cheie trebuie să fie stabilită și folosită pentru o anume perioadă de timp, în esență, un fel de conexiune. De asemenea conexiunile amortizează costurile de inițializare pentru mai multe pachete. În contextul IPsec o "conexiune" este denumită SA (eng.: Security Association, rom.: asociere securizată). O SA este o conexiune simplă între două capete și are asociat un identificator de securitate.

Dacă este nevoie de un trafic securizat în ambele direcții, sunt necesare două asocieri securizate. Identificatorii de securitate sunt transportați în pachetele care se transmit pe aceste conexiuni securizate și sunt folosiți la căutarea cheilor sau a altor informații relevante atunci când sosește un pachet securizat.

Din punct de vedere tehnic, IPsec conține două părți principale. Prima parte descrie două noi antete care pot fi adăugate pachetelor pentru a transporta identificatorul de securitate, datele de control al integrității și alte informații. Cealaltă parte, **ISAKMP** (eng.: **Internet Security Association and Key Managment Protocol**, rom.: **Asociația Securității Internet și Protocolul de Gestiune al Cheilor**) se ocupă cu stabilirea cheilor. Nu vom discuta mai departe despre ISAKMP deoarece (1) este foarte complex și (2) protocolul său principal **IKE** (eng.: **Internet Key Exchange**, rom.: **Schimbul de Chei Internet**), este foarte deficient și trebuie să fie înlocuit (Perlman și Kaufman, 2000).

IPsec poate fi folosit în două moduri. În modul transport, antetul IPsec este inserat chiar după antetul IP. Câmpul de *Protocol* din antetul IP este schimbat pentru a indica faptul că un antet IPsec urmează după antetul normal de IP (înainte de antetul TCP). Antetul IPsec conține informații de securitate, în principal identificatorul SA, un nou număr de secvență și, posibil, o verificare a integrității încărcăturii utile.

În modul tunel, întregul pachet IP, antet și restul, este încapsulat în corpul unui nou pachet IP cu un antet IP complet nou. Modul tunel este folositor când capetele tunelului se termină la o locație diferită de destinația finală. În unele cazuri, capătul tunelului este o mașină poartă de aplicație (eng.: gateway) securizată, de exemplu, de zidul de protecție (eng.: firewall) al unei companii. În acest mod, zidul de protecție încapsulează și decapsulează pachete la trecerea prin el. Terminând tunelul prin această mașină securizată, mașinile din LAN-ul companiei nu trebuie să fie conștiente de IPsec. Doar zidul de protecție trebuie să știe de el.

Modul tunel este de asemenea util când o legătura a unei conexiuni TCP este agregată și se comportă ca un flux criptat, deoarece împiedică un intrus să vadă câte pachete trimite cineva către altcineva. Câteodată doar cunoscând cât de mult trafic este dirijat undeva reprezintă o informație de valoare. De exemplu, dacă în timpul unei crize militare, dimensiunea traficului dintre Pentagon și Casa Albă scade drastic, dar cantitatea de trafic dintre Pentagon și un obiectiv militar aflat în creierul munților Stâncoși ai Colorado-ului crește în aceeași măsură un intrus e capabil să deducă câteva informații utile din aceste date. Studierea șabloanelor de scurgere a pachetelor, chiar dacă ele sunt criptate, se numește **analiză a traficului**. Modul tunel oferă o modalitate de zădărnicire într-o anume măsură. Dezavantajul modului tunel este că adaugă un antet IP în plus, crescând astfel în mod substanțial dimensiunea unui pachet. Prin contrast, modul transport nu afectează atât de mult dimensiunea pachetului.

Primul antet nou este **AH (eng.: Authentication Header, rom.: Antetul de Autentificare)**. El permite controlul integrității și securitate anti-replică, dar nu și confidențialitate (de ex. nu criptează datele). Folosirea AH în modul tunel este ilustrată în fig. 8-27. În IPv4, el este interpus între antetul IP (incluzând orice opțiuni) și antetul TCP. În IPv6 este doar o altă extensie a antetului și este tratată ca atare. De fapt, formatul este apropiat de cel al standardului IPv6 de extensie a antetului. Este posibil ca încărcătura utilă să trebuiască a fie completată până la o anumită lungime pentru algoritmul de autentificare după cum este prezentat în figură.

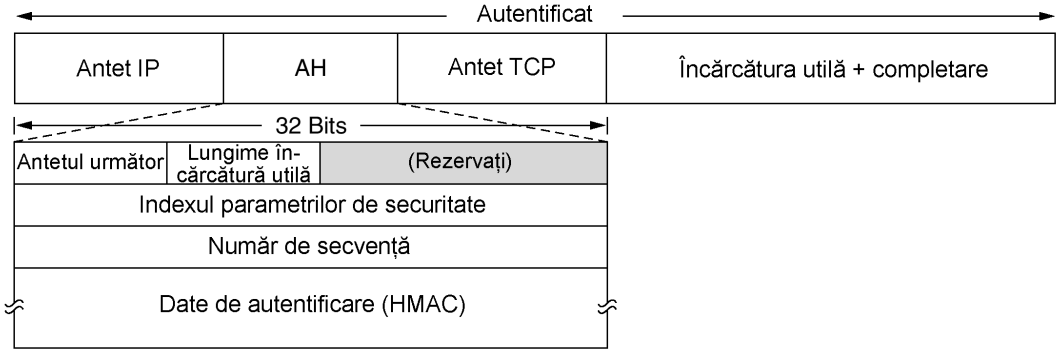


Fig. 8-27. Antetul de autentificare IPsec în modul transport pentru IPv4

Să examinăm acum antetul AH. Câmpul *Antetul următor* este folosit pentru a păstra valoarea anterioară pe care a avut-o câmpul *Protocol* IP înainte de a fi înlocuit cu 51 pentru a indica faptul că urmează un antet AH. În majoritatea cazurilor, aici va fi plasat codul pentru TCP (6) . *Lungimea încărcării utile* o reprezintă numărul de cuvinte de 32 de biți din antetul AH minus 2.

Indexul parametrilor de securitate reprezintă identificatorul de conexiune. Acesta este inserat de către emițător pentru a indica o anumită înregistrare în baza de date a receptorului. Această înregistrare conține cheia partajată folosită în această sesiune și alte informații despre conexiune. Dacă acest protocol ar fi fost inventat de către ITU și nu de către IETF, acest câmp ar fi fost denumit *Număr de circuit virtual*.

Câmpul *Număr de secvență* este folosit pentru a număra toate pachetele trimise pe un SA. Fiecare pachet primește câte un identificator unic, chiar și retransmisiile, cu alte cuvinte copia unui pachet primește un număr diferit de cel original (chiar dacă numărul său de secvență TCP este același). Scopul acestui câmp este de a detecta atacurile prin replică. Aceste numere de secvență nu se pot repeta. Dacă toate cele 2^{32} de numere au fost epuizate, trebuie stabilit un nou SA pentru a continua comunicația.

În sfârșit, câmpul *Date de autentificare* este un câmp de lungime variabilă care conține semnătura digitală. Când este stabilit un SA, cele două părți negociază algoritmul de semnare pe care îl vor folosi. În mod normal, nu este folosită criptografia cu chei publice pentru că pachetele trebuie procesate rapid, iar toți algoritmii cu chei publice sunt prea lenți. Deoarece IPsec este bazat pe criptografia cu chei simetrice iar emițătorul și receptorul negociază o cheie partajată înaintea stabilirii unui SA, cheia partajată este folosită în procesul de semnare. O modalitate simplă este de a calcula rezumatul pentru un pachet și cu cheia partajată. Desigur, cheia partajată nu este transmisă. O schemă ca aceasta este denumită **HMAC (eng. : Hashed Message Authentication Code, rom.: Cod de Autentificare bazat pe un rezumat de mesaj)**. Este mult mai rapid să calculezi un rezumat decât să rulezi întâi SHA-1 și apoi să rulezi RSA asupra rezultatului.

Antetul AH nu permite criptarea datelor, deci este cel mai folosit atunci când este necesară verificarea integrității dar nu este necesară confidențialitatea. O proprietate demnă de notat a antetului AH este aceea că verificarea integrității folosește o parte din câmpurile antetului IP, și anume, acelea care nu se schimbă când pachetul trece de la un ruter la altul. De exemplu, câmpul *Durata de viață* se schimbă la fiecare ruter și nu poate fi inclus în verificarea integrității. În orice caz adresa sursa IP este inclusă în această verificare, un intrus neputând să falsifice originea pachetului.

O altă variantă de antet IPsec este **ESP (eng.: Encapsulating Security Payload, rom.: Încapsularea încărcăturii utile de securitate).** Folosirea acestuia, atât pentru modul transport, cât și pentru modul tunel, este prezentată în fig. 8-28.

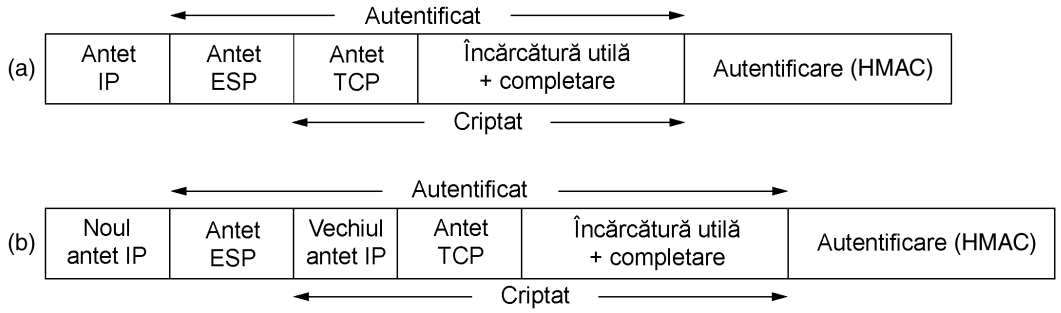


Fig. 8-28. (a) ESP în mod transport (b) ESP în mod tunel

Antetul ESP conține două cuvinte de 32 de biți. Acestea sunt câmpurile *Indexul Parametrilor de Securitate* și, respectiv, *Numărul de Secvență* pe care le-am întâlnit la AH. Un al treilea cuvânt care, în general, le urmează (dar care din punct de vedere tehnic nu este parte a antetului) este câmpul *Vector de Inițializare*, folosit pentru criptarea datelor; dacă nu se folosește criptarea acest câmp este omis.

De asemenea, ESP oferă verificarea integrității pentru HMAC, așa cum o face și AH, dar în loc să fie inclus în antet, urmează după încărcătura utilă, așa cum se arată în fig. 8-28. Includerea HMAC la sfârșit are un avantaj în implementarea hardware. HMAC poate fi calculat pe măsură ce biții ies dintr-o interfață de rețea atașarea sa la sfârșit fiind foarte simplă. Din această cauză Ethernet și alte LAN-uri au propriile CRC la sfârșitul pachetelor în loc să le aibă la începutul acestora. Cu AH, pachetul trebuie să fie păstrat într-o zonă tampon și semnătura să fie calculată înainte ca pachetul să fie trimis, reducând astfel numărul de pachete care pot fi transmise pe secundă.

Deoarece ESP poate face tot ceea ce poate face AH și chiar mai mult și faptul că este mai eficient la inițializare decât acesta, se ridică întrebarea: De ce să ne mai complicăm cu AH? Răspunsul este mai mult istoric. La început, AH se ocupa numai de integritate iar ESP numai de confidențialitate. Mai târziu, integritatea a fost adăugată și la ESP, dar cei care au proiectat AH nu au vrut ca acesta să dispară după ce au muncit atât de mult la el. În orice caz, singurul lor argument real este faptul că AH verifică o parte a antetului IP, ceea ce ESP nu face; dar acesta este un argument nesemnificativ. Un alt argument nesemnificativ este acela că un produs care suportă AH, dar nu suportă ESP, poate avea mai puține probleme în a obține licența de export, datorită faptului că nu folosește criptarea. Este posibil ca în viitor AH să nu mai fie folosit.

8.6.2 Ziduri de protecție

Posibilitatea de a conecta orice calculator, de oriunde, cu orice alt calculator, de oriunde, este o sabie cu două tăișuri. Pentru persoanele aflate acasă, colindatul prin Internet aduce multe bucurii. Pentru administratorii pe probleme de securitate ai firmelor, este un coșmar. Multe companii au mari cantități de informație confidențială sub formă electronică - secrete de afaceri, planuri de dezvoltare produse, strategii de marketing, analize financiare etc. Dezvăluirea acestor informații către un competitor poate avea consecințe cumplite.

În afara pericolului scurgerii de informații, există și un pericol al infiltrării de informații. În particular, virușii, viermii și alți dăunători digitali pot încălca securitatea, distruge informații de valoare și irosi o mare cantitate din timpul administratorilor care încearcă să curețe dezordinea pe care o lasă. Deseori ei sunt importați de angajați neglijenți care vor să joace un joc nou, grozav.

În consecință, sunt necesare mecanisme pentru a păstra biții „buni" în interior și biții „răi" afară. O metodă este folosirea IPsec. Această abordare protejează datele în tranzit între situri sigure. Cu toate acestea, IPsec nu face nimic pentru a ține dăunătorii digitali și spărgătorii în afara LAN-ului companiei. Pentru a realiza acest obiectiv, este nevoie să studiem zidurile de protecție (eng.: firewalls).

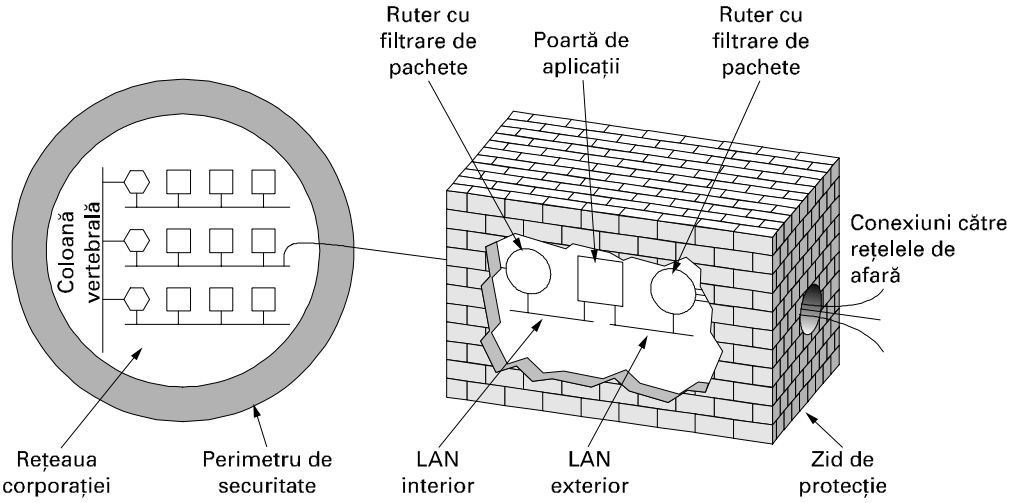


Fig. 8-29. Un zid de protecție format din două filtre de pachete și o poartă de aplicații.

Zidurile de protecție sunt doar o adaptare modernă a acelei vechi soluții de securitate medievală: săparea unui șanț adânc de apărare în jurul castelului dvs. Acest proiect forța pe oricine dorea să intre sau să iasă din castel să treacă peste un singur pod mobil, unde puteau fi inspectați de poliția de intrare/ieșire. Cu rețelele se poate face același truc: o companie poate avea multe LAN-uri conectate în moduri arbitrare, dar tot traficul către sau de la companie este forțat printr-un pod mobil electronic (zidul de protecție), așa cum este prezentat în fig. 8-29.

În această configurație zidul de protecție are două componente: două rutere care fac filtrare de pachete și o poartă de aplicații. Există, de asemenea, configurații și mai simple, dar avantajul acestui proiect este că fiecare pachet trebuie să tranziteze două filtre și o poartă de aplicație pentru a intra sau ieși. Nu există altă rută. Este destul de clar că cititorii care consideră că un singur punct de verificare a securității este suficient nu au făcut recent un zbor internațional cu o companie aeriană.

Fiecare **filtru de pachete** este un ruter standard echipat cu unele funcții suplimentare. Acestea permit inspectarea fiecărui pachet care intră sau iese. Pachetele care îndeplinesc anumite criterii sunt transmise normal. Cele care nu trec testul sunt eliminate.

În fig. 8-29, este foarte probabil ca filtrul de pachete din interiorul LAN-ului să verifice pachetele care ies, iar cel din exteriorul LAN-ului să verifice pachetele care intră. Pachetele care trec de prima barieră merg la poarta de aplicații pentru o examinare ulterioară. Motivul introducerii a două filtre

de pachete în rețele diferite este de a asigura că nici un pachet nu intră sau iese fără a fi obligat să treacă prin poarta de aplicații: nu există nici o cale pe care să o ocolească.

Filtrele de pachete sunt, în mod tipic, dirijate de tabele configurate de administratorul de sistem. Aceste tabele enumeră sursele și destinațiile acceptabile, sursele și destinațiile care sunt blocate și reguli implicite despre ce se face cu pachetele care vin sau se duc la alte mașini.

În cazul uzual al configurării TCP/IP, o sursă și o destinație constau dintr-o adresă IP și un port. Porturile indică ce serviciu este dorit. De exemplu, portul 23 TCP este pentru Telnet, portul 79 TCP este pentru Finger, iar portul 119 TCP este pentru știrile USENET. O companie poate bloca toate pachetele de intrare pentru toate adresele IP asociate cu unul din aceste porturi. În acest mod, nimeni din afara companiei nu se poate conecta prin telnet, sau să caute persoane folosind demonul de Finger. Mai mult, compania va fi scutită ca angajații să-și petreacă toată ziua citind știri USENET.

Blocarea pachetelor care ies este mai complicată, deoarece, deși cele mai multe situri aderă la convențiile standard de numire a porturilor, nu sunt obligate să o facă. Mai mult, pentru servicii importante, cum ar fi FTP (File Transfer Protocol - protocol de transfer fișiere), numerele de port sunt atribuite dinamic. În plus, deși blocarea conexiunilor TCP este dificilă, blocarea pachetelor UDP este și mai grea datorită faptului că se știe foarte puțin a priori despre ce vor face. Multe filtre de pachete pur și simplu interzic în totalitate traficul UDP.

A doua jumătate a mecanismului de zid de protecție este **poarta de aplicație**. În loc să trateze pachete brute, o poartă operează la nivelul aplicație. O poartă de poștă electronică, de exemplu, poate fi configurată să examineze fiecare mesaj care intră sau iese. Pentru fiecare mesaj, ea ia decizia de a-l transmite sau elimina pe baza câmpurilor din antet, a dimensiunii mesajului sau chiar a conținutului (de exemplu, la o instalație militară, prezența cuvintelor ca „nuclear” sau „bombă” pot cauza generarea unor acțiuni speciale).

Proiectele de instalare au libertatea de a configura una sau mai multe porți de aplicație pentru aplicații specifice, dar nu este ieșit din comun ca organizații suspicioase să permită intrarea și ieșirea poștei electronice și, probabil, folosirea World Wide Web, dar să interzică orice altceva ca fiind prea riscant. Combinat cu criptarea și cu filtrarea de pachete, acest aranjament oferă o cantitate limitată de securitate cu costul unor inconveniente.

Chiar dacă zidul de protecție este configurat perfect, există încă o groază de probleme de securitate. De exemplu, dacă un zid de protecție este configurat să accepte pachete doar de la anumite rețele (de ex. alte sedii ale companiei), un intrus din exteriorul zidului de protecție poate să-și pună o adresă falsă pentru a evita această verificare. Dacă un individ din interior dorește să vândă documentele secrete, el le poate cripta sau chiar fotografia și apoi să sustragă pozele ca fișiere JPEG, care evită orice filtru de texte.

Și încă nu am discutat faptul că 70% din totalul atacurilor vin din interiorul rețelei protejate de zidul de protecție, de exemplu, de la angajații nemulțumiți (Schneier, 2000).

În plus, există o întreagă altă clasă de atacuri cărora zidurile de protecție nu le pot face față. Ideea de bază a unui zid de protecție este de a împiedica intrușii să pătrundă în rețeaua protejată și de a împiedica datele secrete să iasă din acea rețea. Din păcate, există oameni care nu au altceva mai bun de făcut, decât să încerce să scoată din funcțiune anumite situri. Ei realizează acest lucru prin trimiterea unui număr foarte mare de pachete legitime către o țintă, până când aceasta va fi scoasă din funcțiune datorită încărcării mari. De exemplu, pentru a distruge un sit de web, un intrus poate trimite un pachet TCP SYN pentru a stabili o conexiune. Ca urmare, situl va aloca, într-o tabelă, o intrare pentru acea conexiune și va trimite ca răspuns un pachet SYN +ACK. Dacă intrusul nu răspunde intrarea din tabelă va fi ocupată pentru un interval de timp de până la câteva secunde, până

când se va produce un timeout. Dacă intrusul va trimite mii de cereri de conexiune, toate intrările din tabelă vor fi ocupate astfel încât nu vor mai putea fi stabilite noi conexiuni legitime. Atacurile în care scopul intrusului este de a opri funcționarea țintei, în locul sustragerii de informații, sunt denumite atacuri **DoS (eng.: Denial of Service, rom.: Refuzul Serviciilor)**. În mod obișnuit, pachetele de cereri au o adresă sursă falsă pentru ca intrusul să nu poată fi detectat ușor.

O variantă și mai pesimistă este aceea în care intrusul a pătruns deja în sute de calculatoare aflate în alte zone ale lumii, pe care apoi le comandă să atace aceeași țintă în același timp. Nu numai că această abordare crește forța de atac a intrusului, dar ea reduce șansele ca ea să fie detectată, deoarece pachetele provin de la un mare număr de mașini aparținând unor utilizatori ce nu sunt suspecți. Un astfel de atac este denumit atac **DDoS (eng.: Distributed Denial of Service, rom.: Refuzul serviciilor realizat în mod distribuit)**. Împotriva acestui atac este greu de găsit o apărare. Chiar dacă mașina atacată poate recunoaște rapid o cerere falsă, trece un anumit timp pentru procesarea și ignorarea acestei cereri, iar dacă sosesc destul de multe cereri pe secundă procesorul va fi ocupat tot timpul cu tratarea acestora.

8.6.3 Rețele private virtuale

Multe companii au birouri răspândite în mai multe orașe și uneori în mai multe țări. În trecut, înaintea apariției rețelelor de date publice, era obișnuită închirierea de către aceste companii a unor linii telefonice, aparținând companiilor de telefonie, între unele sau între toate perechile de locații ale birourilor. Unele companii încă mai fac acest lucru. O rețea alcătuită din calculatoarele unei companii și liniile telefonice închiriate este denumită **rețea privată**. Un exemplu de rețea privată conectând trei birouri este arătată în fig. 8-30(a).

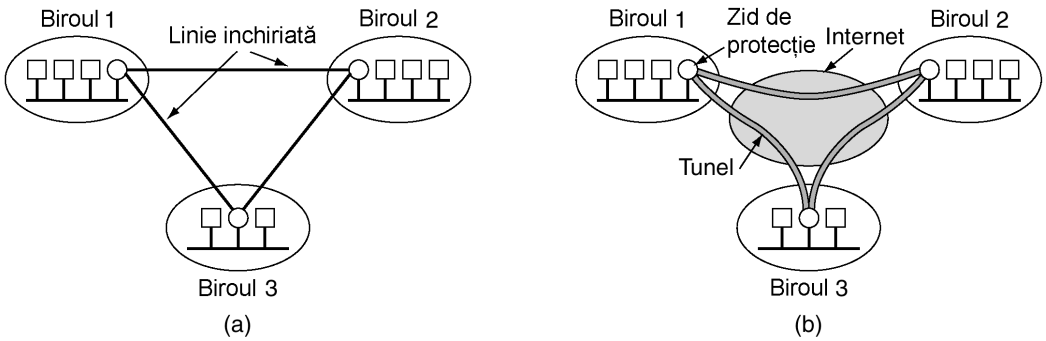


Fig. 8-30. (a) O rețea privată folosind linii închiriate (b) O rețea privată virtuală

Rețelele private funcționează bine și sunt foarte sigure. Dacă singurele linii disponibile sunt linii închiriate, atunci nu există trafic care să se scurgă în afara companiei, iar intrușii trebuie să se conecteze fizic la liniile respective pentru a pătrunde în rețea, ceea ce nu este un lucru ușor de realizat. Problema care apare cu rețelele private este acea că închirierea unei singure linii T1 costă mii de dolari pe lună, iar liniile T3 sunt de câteva ori mai scumpe. Când, mai târziu, au apărut rețelele de date publice și Internetul, multe companii au vrut să-și transmită traficul de date (și posibil cel de voce) prin rețelele publice, dar fără a renunța la securitatea unei rețele private.

Această necesitate a condus în curând la inventarea rețelelor private virtuale - **VPN (eng.: Virtual Private Networks, rom.: rețele private virtuale),** care sunt rețele construite deasupra unor rețele pu-

blice, dar care beneficiază de proprietățile unei rețele private. Aceste rețele sunt denumite virtuale pentru că ele constituie o iluzie, așa cum circuitele virtuale nu sunt circuite reale și cum memoria virtuală nu este o memorie reală.

Deși VPN-urile pot fi implementate peste ATM (sau frame relay), o tendință populară aflată în creștere este de a construi VPN-uri direct peste Internet. O modalitate de proiectare obișnuită este de a echipa fiecare birou cu un zid de protecție și de a crea tuneluri prin Internet între toate perechile de birouri, după cum este ilustrat în fig. 8-30(b). Dacă pentru tunelare este folosit IPsec, atunci este posibilă agregarea întregului trafic dintre oricare două perechi de birouri într-un singur SA autentificat și criptat, oferindu-se astfel controlul integrității, confidențialitate și chiar o imunitate considerabilă asigurată analizei de trafic.

Când un sistem este pornit, fiecare pereche de ziduri de protecție trebuie să negocieze parametrii SA-ului ei, incluzând serviciile, modurile, algoritmii și cheile. Multe ziduri de protecție au înglobate capabilități de VPN, deși chiar și unele rutere obișnuite beneficiază de acestea. Dar, deoarece zidurile de protecție au prioritate în problema securității, este natural a avea tuneluri care încep și se termină în ziduri de protecție, furnizând o separare clară între companie și Internet. Astfel, zidurile de protecție, VPN-urile și IPsec folosit cu ESP în mod tunel formează o combinație naturală și larg folosită în practică.

Odată ce a fost stabilit un SA, transferul datelor poate începe. Pentru un ruter din Internet, un pachet care traversează un tunel VPN este un pachet obișnuit. Singurul lucru neobișnuit la acesta este prezența antetului IPsec după antetul IP, dar deoarece aceste extra-antete nu au efect în procesul de rutare, ruterele nu le iau în considerare.

Un avantaj cheie al organizării VPN în acest fel este transparența completă pentru programele utilizatorilor. Zidurile de protecție inițializează și gestionează SA-urile. Singura persoană care are cunoștință de acest proces este administratorul de sistem care configurează și gestionează zidul de protecție. Pentru oricine altcineva lucrurile arată ca și cum ar fi o rețea privată bazată pe linie închiriată. Pentru mai multe despre VPN consultați (Brown, 1999 și Izzo, 2000).

8.6.4 Securitatea în comunicațiile fără fir

Este surprinzător de ușor de proiectat un sistem care din punct de vedere logic, este complet securizat folosind VPN și ziduri de protecție, dar prin care, în practică, informația se scurge ca prin sită. Această situație poate fi întâlnită dacă o parte dintre mașini sunt fără fir și folosesc comunicația radio, care trece prin zidul de protecție în ambele sensuri. Aria de acoperire a rețelelor 802.11 este foarte adesea de câteva sute de metri, astfel că oricine dorește să spioneze o companie, poate să vină dimineața în parcarea pentru angajați, să lase un calculator portabil care este configurat pentru o rețea 802.11 să înregistreze tot ce aude și să dispară pentru restul zilei. Până după-amiaza târziu, discul dur va fi plin de lucruri valoroase. Teoretic, această scurgere nu ar trebui să aibă loc. Teoretic, nici lumea nu ar trebui să jefuiască bănci.

O mare parte a problemei de securitate poate fi urmărită până la producătorii de stații de bază care comunică fără fir (puncte de acces), care încearcă să-și facă produsele prietenoase pentru utilizatori. De obicei, dacă utilizatorul scoate dispozitivul din cutie și îl introduce în priză, acesta începe să funcționeze imediat – aproape fără nici un fel de securitate, împrăștiind secrete către oricine se află în aria de acoperire radio. Dacă apoi este și legat la o rețea Ethernet, tot traficul Ethernet va apărea dintr-o dată și în locul de parcare. Rețelele fără fir sunt visul oricărui agent ascuns: date gratuite, fără să trebuiască să lucrezi pentru ele. Prin urmare, nici nu mai trebuie spus că securitatea

este și mai importantă pentru rețelele fără fir decât pentru cele cu fir. În această secțiune ne vom uita la câteva moduri în care rețelele fără fir tratează securitatea. Câteva informații suplimentare pot fi găsite în (Nicholas și Lekkas, 2002).

Securitatea 802.11

Standardul 802.11 recomandă un protocol de securitate la nivelul legătură de date care se numește **WEP** (eng.: **Wired Equivalent Privacy** rom.: **Confidențialitate Echivalentă cu cea Cablată**), care este proiectat pentru a face securitatea unui LAN cu comunicație fără fir la fel de bună ca cea a unui LAN cablat. Cum LAN-urile cu cabluri nu au de obicei nici un fel de securitate, acest obiectiv este ușor de îndeplinit și WEP îl îndeplinește, după cum vom vedea.

Când securitatea 802.11 este activată, fiecare stație are o cheie secretă comună cu stația de bază. Standardul nu specifică cum sunt distribuite cheile. Pot fi încărcate înainte de către producător. Pot fi interschimbate înainte prin rețeaua cu fire. În fine, fie stația de bază, fie mașina utilizatorului poate alege o cheie aleatoare și o poate trimite prin aer criptată cu cheia publică a celeilalte stații. Odată stabilită, cheia rămâne neschimbată în general pentru luni sau ani.

Criptarea WEP folosește un cifru flux bazat pe algoritmul RC4. RC4 a fost proiectat de Ronald Rivest și a fost ținut secret până în 1994 când s-au scurs niște informații și a fost publicat pe Internet. După cum am mai subliniat înainte, este aproape imposibil ca algoritmii să fie păstrați secreți, chiar dacă scopul este protejarea proprietății intelectuale (cum a fost în acest caz), mai degrabă decât securitatea prin obscuritate (care nu a fost scopul cu RC4). În WEP, RC4 generează un șir-cheie (eng.: keystream) care este combinat prin XOR cu textul clar pentru a forma textul cifrat.

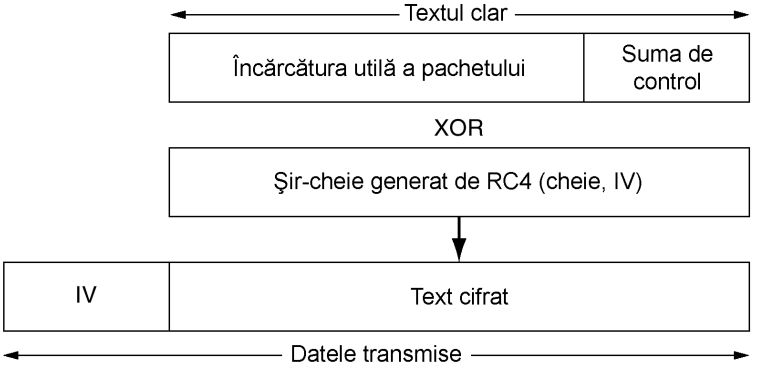


Fig. 8-31. Criptarea pachetelor folosind WEP

Încărcătura utilă a fiecărui pachet este criptată folosind metoda din fig. 8-31. Mai întâi, încărcătura este verificată folosind polinomul CRC-32, iar suma de control este adăugată încărcăturii pentru a forma textul clar pentru algoritmul de criptare. Acest text clar este combinat XOR cu o bucată din șirul cheie de aceeași lungime. Rezultatul este textul cifrat. IV-ul utilizat pentru pornirea RC4 este trimis împreună cu textul cifrat. Când receptorul primește pachetul, extrage din acesta încărcătura utilă criptată, generează șirul cheie din cheia secretă și din IV pe care tocmai l-a primit și apoi efectuează operația XOR între șirul cheie și încărcătura utilă a pachetului pentru a obține textul în clar. Apoi poate verifica suma de control pentru a se asigura că pachetul a ajuns intact.

Deși această abordare arată bine la prima vedere, a fost deja publicată o metodă de a o sparge (Borisov et. al, 2001). Mai jos vom rezuma rezultatele ei. În primul rând, surprinzător de multe

instalații folosesc aceeași cheie comună pentru toți utilizatorii, caz în care fiecare utilizator poate citi traficul tuturor celorlalți utilizatori. Acest lucru este desigur echivalent cu Ethernet-ul, dar nu este foarte sigur.

WEP poate fi atacat chiar dacă fiecare utilizator are o cheie distinctă,. Din moment ce cheile sunt în general stabile pentru perioade lungi de timp, standardul WEP recomandă (dar nu impune) ca IV-ul să fie schimbat la fiecare pachet pentru a evita atacul de tip reutilizare asupra șirului cheie, atac discutat în secțiunea 8.2.3. Din nefericire, multe plăci 802.11 pentru calculatoarele portabile resetează IV la 0 când placa este introdusă în calculator și apoi îl incrementează cu 1 la fiecare pachet trimis. Cum utilizatorii scot și apoi reinserează frecvent aceste plăci, pachetele cu valori mici pentru IV sunt destul de obișnuite. Dacă Trudy poate să colecteze mai multe pachete trimise de același utilizator, care au aceeași valoare pentru IV (care este trimis în clar cu fiecare pachet), atunci ea poate să calculeze XOR între două valori de textul clar și poate astfel să spargă cifrul.

Dar, chiar dacă placa 802.11 alege o valoare aleatoare pentru fiecare pachet, IV are doar 24 de biți, astfel încât după ce au fost trimise 2^{24} pachete, va trebui ca valorile să fie refolosite. Mai rău, folosind valori aleatoare pentru IV, numărul probabil de pachete care trebuie trimise înainte ca același număr să fie folosit de două ori este în jur de 5000, datorită atacului de tip „ziua de naștere” descris în secțiunea 8.4.4. Ca urmare, dacă Trudy ascultă pentru câteva minute, ea este aproape sigură că va captura două pachete cu același IV și aceeași cheie. Efectuând operația XOR între cele două texte cifrate, ea este capabilă să obțină combinația XOR dintre textele în clar. Această secvență de biți poate fi atacată în diferite moduri pentru a descoperi textele în clar. Cu mai multă muncă, șirul cheie pentru acel IV poate fi obținut de asemenea. Trudy poate continua să lucreze în acest mod pentru a realiza un dicționar de șiruri cheie pentru diferite IV. Odată spart un IV, toate pachetele trimise cu acesta în viitor (dar și în trecut) pot fi complet decriptate.

Mai mult, deoarece IV-rile sunt folosite aleator, odată ce Trudy a determinat o pereche (IV, șir-cheie) validă, ea o poate folosi pentru a genera toate pachetele pe care le dorește și astfel, să intervină activ în comunicație. Teoretic, un receptor ar putea observa că dintr-o dată un număr mare de pachete au toate același IV, dar (1) WEP permite acest lucru și (2) oricum nimeni nu verifică acest lucru.

În sfârșit, CRC-ul nu valorează prea mult, deoarece este posibil ca Trudy să modifice încărcătura utilă și apoi să facă schimbările corespunzătoare în CRC, fără a trebui măcar să elimine criptarea. Pe scurt, spargerea securității 802.11 este destul de evidentă, și nici măcar nu am enumerat toate atacurile pe care le-au găsit Borisov et. al.

În august 2001, la o lună după ce a fost prezentată lucrarea lui Borisov et. al, a fost publicat un alt atac devastator asupra WEP (Fluhrer et. al, 2001). Acesta a găsit slăbiciuni criptografice chiar în RC4. Fluhrer et. al au descoperit că multe dintre chei au proprietatea că este posibilă deducerea câtorva dintre biții cheii din șirul-cheie. Dacă acest atac este pus în aplicare repetat, este posibilă deducerea întregii chei cu un efort modest. Fiind înclinați mai mult către teorie, Fluhrer et. al nu au încercat efectiv să spargă vreun LAN 802.11.

În contrast, când un student la cursurile de vară și doi cercetători de la AT&T Labs au aflat despre atacul lui Fluhrer et. al, s-au decis să îl încerce într-un caz real (Stubblefield et. al, 2002). Într-o săptămână ei au spart prima lor cheie de 128 de biți dintr-un LAN 802.11 de producție și cea mai mare parte a săptămânii a fost de fapt dedicată căutării celei mai ieftine plăci de rețea 802.11, obținerii permisiunii de a o cumpăra, instalării și testării ei. Programarea a durat efectiv doar două ore.

Când și-au anunțat rezultatele, CNN a emis o știre intitulată „Un spărgător de rutină sparge criptarea comunicațiilor fără fir”, în care niște guru din industrie au încercat să minimalizeze rezultatele lor, spunând că ceea ce au făcut ei a fost trivial, date fiind rezultatele lui Fluhrer et. al Chiar dacă

această observație este tehnic adevărată, rămâne faptul că eforturile combinate ale acestor două echipe au demonstrat o scăpare fatală în WEP și 802.11.

În 7 Septembrie, 2001, IEEE a răspuns faptului că WEP era atunci complet spartă printr-o declarație scurtă cu șase puncte care pot fi rezumate grosier în felul următor:

1. Noi v-am spus că securitatea WEP nu era mai bună decât cea a Ethernet-ului.
2. O amenințare mult mai mare este să uiți complet să activezi securitatea.
3. Încercați să folosiți altă securitate (de ex., securitatea nivelului transport)
4. Versiunea următoare, 802.11i, va avea o securitate mai bună.
5. Certificările ulterioare vor impune folosirea 802.11i.
6. Vom încerca să ne dăm seama ce se poate face până când apare 802.11i.

Am parcurs în detaliu toată această poveste pentru a demonstra faptul că o securitate solidă nu este ușor de obținut, nici pentru experți.

Securitatea Bluetooth

Bluetooth are o arie de acoperire mult mai mică decât 802.11 și nu poate fi atacată din locul de parcare, dar securitatea este și aici o problemă. De exemplu, imaginați-vă calculatorul lui Alice care are o tastatură fără fir Bluetooth. În absența securității, dacă Trudy este întâmplător în biroul de alături, ea ar putea citi tot ce scrie Alice, inclusiv toată corespondența electronică trimisă. De asemenea, ea ar putea captura tot ce trimite calculatorul lui Alice imprimantei Bluetooth care se află lângă ea (corespondența electronică primită și rapoarte confidențiale). Din fericire, Bluetooth are o schemă de securitate pentru a încerca să le învingă pe orice Trudy din lume. Vom rezuma în continuare principalele trăsături ale acesteia.

Bluetooth are trei moduri de securitate, de la nici o securitate la criptarea tuturor datelor și controlul integrității. La fel ca și 802.11, dacă securitatea este dezactivată (opțiunea care este selectată inițial) atunci nu există securitate. Majoritatea utilizatorilor au securitatea dezactivată până când are loc o spargere; apoi o activează. În lumea agriculturii, această abordare este cunoscută drept încuierea ușii hambarului după ce a fugit calul.

Bluetooth oferă securitate pe mai multe niveluri. La nivelul fizic, salturile în frecvență oferă un pic de securitate, dar deoarece oricare dispozitiv Bluetooth care se mișcă într-un piconet trebuie informat de secvența de salturi în frecvență, această secvență evident că nu este un secret. Adevărata securitate începe când un sclav (eng.: slave) nou-venit cere un canal cu stăpânul (eng.: master). Este de presupus că cele două dispozitive împart o cheie secretă stabilită anterior. În unele cazuri, cheia este fixată în ambele dispozitive de către producător (de ex., pentru un set de căști cu microfon și un telefon mobil vândute împreună). În alte cazuri, un dispozitiv (de ex., setul de căști) are o cheie fixă, iar utilizatorul trebuie să introducă acea cheie în celălalt dispozitiv (de ex., telefonul mobil) ca un număr zecimal. Aceste chei comune se numesc **chei de trecere** (eng.: **passkeys**).

Pentru a stabili un canal, sclavul și stăpânul verifică fiecare dacă celălalt cunoaște cheia de trecere. Dacă da, ei negociază dacă acel canal va fi criptat, cu controlul integrității, sau amândouă. Apoi ei aleg o cheie de sesiune aleatoare de 128 de biți, dintre care câțiva ar putea fi publici. Ideea acestei slăbiri a cheii este alinierea la restricțiile guvernamentale din diferite țări proiectate, pentru a preveni exportul, sau utilizarea cheilor mai lungi decât acelea pe care guvernul le poate sparge.

Criptarea folosește un cifru-flux denumit $\mathbf{E_0}$; controlul integrității folosește **SAFER+**. Amândouă sunt cifruri bloc tradiționale cu chei simetrice. SAFER+ a fost înscris în concursul AES, dar a fost eliminat în prima rundă pentru că era mai lent decât ceilalți candidați. Bluetooth a fost finalizat înainte ca cifrul AES să fie ales; altfel, foarte probabil, ar fi folosit Rijndael.

Criptarea actuală folosind cifrul-flux este prezentată în fig. 8-14, cu textul în clar combinat prin XOR cu șirul-cheie pentru a genera textul cifrat. Din nefericire, E_0 însuși (ca și RC4) ar putea avea slăbiciuni fatale (Jakobsson și Wetzel, 2001). Deși nu a fost spart până la momentul scrierii acestei cărți, similitudinile lui cu cifrul A5/1, al cărui eșec spectaculos compromite tot traficul telefonic GSM, sunt o cauză de îngrijorare (Biryukov et. al, 2000). Uneori este uimitor pentru unii (incluzând autorul) că în perenul joc de-a șoarecele și pisica între criptografi și criptanaliști, criptanaliști sunt atât de des în partea învingătoare.

O altă problemă de securitate este că Bluetooth autentifică doar dispozitivele, nu utilizatorii, astfel că furtul unui dispozitiv Bluetooth poate permite hoțului accesul la contul financiar și la alte conturi ale utilizatorului. Oricum, Bluetooth implementează de asemenea securitatea pentru nivelele înalte, astfel că în cazul unei spargeri a securității de la nivelul legătură, securitatea se menține, în special pentru aplicațiile care necesită introducerea manuală a unui cod de PIN de la un fel de tastatură pentru a completa tranzacția.

Securitatea WAP 2.0

În mare parte, Forumul WAP a învățat ceva din greșeala de a avea o stivă de protocoale nestandard în WAP 1.0. WAP 2.0 folosește generos protocoalele standard la toate nivelurile. Securitatea nu este o excepție. Deoarece este bazat pe IP, el suportă folosirea integrală a IPsec la nivelul rețea. La nivelul transport, conexiunile TCP pot fi protejate de TLS, un standard IETF pe care îl vom studia mai târziu în acest capitol. Și mai sus, folosește autentificarea HTTP a clientului, așa cum e definită în RFC 2617. Bibliotecile criptografice de la nivelul aplicație oferă controlul integrității și nerepudierea. Una peste alta, din moment ce WAP 2.0 este bazat pe standarde bine cunoscute, există o șansă ca serviciile sale de securitate, în particular, confidențialitatea, controlul integrității și nerepudierea să valoreze mai mult decât securitatea 802.11 și Bluetooth.

8.7 PROTOCOALE DE AUTENTIFICARE

Autentificarea (eng.: *authentication*) este tehnica prin care un proces verifică dacă partenerul său de comunicație este cel presupus a fi și nu un impostor. Verificarea identității unui proces de la distanță, în cazul unui intrus activ și răuvoitor, este surprinzător de dificilă și necesită protocoale complexe bazate pe criptografie. În această secțiune, vom studia câteva dintre numeroasele protocoale de autentificare folosite în rețelele nesigure de calculatoare.

Ca fapt divers, anumiți oameni confundă autorizarea cu autentificarea. Autentificarea se ocupă cu problema de a ști dacă într-adevăr comunici cu un anumit proces. Autorizarea se ocupă cu ceea ce îi este permis unui proces să facă. De exemplu, un proces client contactează un server de fișiere și spune: „Eu sunt procesul lui Scott și vreau să șterg fișierul *cookbook.old*”. Din punctul de vedere al serverului de fișiere, trebuie găsite răspunsurile la două întrebări:

1. Chiar este procesul lui Scott? (autentificare)
2. Are Scott permisiunea de a șterge *cookbook.old*? (autorizare)

Doar după ce s-a răspuns afirmativ, fără ambiguități, la ambele întrebări poate avea loc acțiunea cerută. De fapt prima dintre cele două este cu adevărat întrebarea cheie. Odată ce serverul de fișiere

știe cu cine vorbește, verificarea autorizării este doar o problemă de căutare în intrările tabelelor și bazelor de date locale. Din acest motiv, în această secțiune ne vom concentra asupra autentificării.

Modelul general pe care îl folosesc toate protocoalele de autentificare este următorul. Alice începe prin a trimite un mesaj fie lui Bob, fie unui centru autorizat de distribuire a cheilor **KDC** (eng.: **Key Distribution Center**), care este presupus a fi credibil. Urmează alte câteva schimburi de mesaje în diferite direcții. În timp ce aceste mesaje sunt transmise, Trudy, le poate intercepta, modifica sau retrimite în scopul de a-i înșela pe Alice și Bob sau doar pentru a încurca lucrurile. Cu toate acestea, la încheierea protocolului de comunicare, Alice este sigură că a vorbit cu Bob, iar acesta este sigur că a vorbit cu Alice. Mai mult decât atât, în cele mai multe protocoale, cei doi vor fi stabilit și o **cheie secretă de sesiune**, pentru folosirea în conversațiile viitoare. În practică, din motive de performanță, tot traficul de date este criptat folosind criptografia cu cheie secretă (de obicei AES sau DES triplu), în timp ce criptografia cu cheie publică este larg folosită în protocoale de autentificare și pentru stabilirea unei chei de sesiune.

Motivul pentru utilizarea unei chei de sesiune noi, alese aleator, pentru fiecare conexiune, este minimizarea mărimii traficului care este transmis cu cheile secrete sau publice ale utilizatorilor, pentru a reduce cantitatea de text cifrat pe care intrusul o poate obține și pentru a reduce pagubele care se pot produce dacă un proces eșuează, iar vidajul său de memorie (eng.: core dump) cade în mâini rele. Din fericire, singura cheie prezentă va fi cheia de sesiune. Toate cheile permanente trebuie să fie anulate cu grijă după stabilirea sesiunii.

8.7.1 Autentificare bazată pe cheie secretă partajată

Pentru primul nostru protocol de autentificare, vom presupune că Alice și Bob partajează deja o cheie secretă, K_{AB}. Această cheie partajată poate să fi fost stabilită anterior fie prin telefon, fie direct, dar în nici un caz nu prin intermediul unei rețele (nesigure).

Protocolul este bazat pe un principiu ce poate fi întâlnit în multe protocoale de autentificare: o parte trimite un număr aleatoriu celeilalte, care îl transformă apoi într-un anumit mod și returnează rezultatul. Astfel de protocoale se numesc protocoale **provocare-răspuns** (eng.: *challenge-response*). În acest protocol și în cele ce urmează va fi folosită următoarea notație:

A, B reprezintă identitățile lui Alice și Bob.
R_i reprezintă provocările, unde indicele identifică pe cel ce trimite provocarea.
K_i sunt cheile, unde i indică proprietarul.
K_S este cheia de sesiune.

Secvența de mesaje pentru primul nostru protocol de autentificare cu cheie partajată este ilustrată în fig. 8-32. În mesajul 1 Alice îi trimite lui Bob identitatea sa, A, într-un mod pe care Bob îl înțelege. Bineînțeles că Bob nu are posibilitatea de a ști dacă acest mesaj vine de la Alice sau de la Trudy, astfel încât el alege o provocare, un număr aleator mare, R_B, și îl trimite înapoi spre „Alice" sub forma mesajului 2, ca text clar. Alice criptează atunci mesajul cu cheia pe care ea o partajează cu Bob și trimite textul cifrat $K_{AB}(R_B)$ înapoi, ca mesajul 3. Când Bob vede acest mesaj el știe imediat că vine de la Alice, deoarece Trudy nu cunoaște K_{AB} și astfel nu putea să genereze mesajul respectiv. Mai mult decât atât, deoarece R_B a fost ales aleator dintr-un spațiu mare (să spunem, de exemplu, numere aleatoare pe 128 de biți), este destul de puțin probabil ca Trudy să fi văzut R_B și răspunsul asociat lui într-o sesiune anterioară. Este la fel de puțin probabil ca ea să fi putut ghici răspunsul corect la orice provocare.

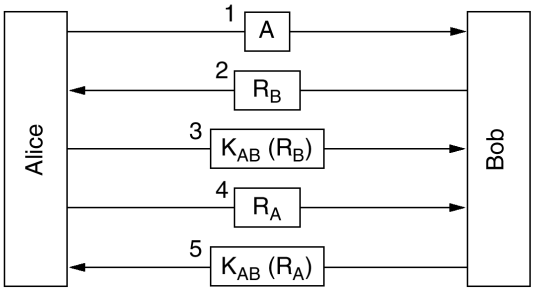


Fig. 8-32. Autentificare în doi pași folosind un protocol de tipul provocare-răspuns.

În acest moment, Bob este sigur că vorbește cu Alice, dar Alice nu este sigură de nimic. Din punctul ei de vedere, s-ar putea ca Trudy să fi interceptat mesajul 1 și să fi trimis înapoi replica R_B. Poate că Bob a murit azi-noapte. Pentru a descoperi cu cine vorbește, Alice alege un număr aleator R_A și îl trimite lui Bob ca text clar, în mesajul 4. Când Bob răspunde cu $K_{AB}(R_A)$, Alice știe că vorbește cu Bob. Dacă ei vor să stabilească acum o cheie de sesiune, Alice poate alege una, K_S, și o poate trimite lui Bob criptată cu K_{AB}.

Protocolul din fig. 8-32 conține cinci mesaje. Să vedem dacă putem să fim isteți și să eliminăm o parte din ele. O variantă este ilustrată în fig. 8-33. Aici Alice inițiază protocolul provocare-răspuns, în loc de a-l aștepta pe Bob să o facă. Similar, în timp ce răspunde la provocarea lui Alice, Bob o trimite pe a sa. Întregul protocol poate fi redus la trei mesaje în loc de cinci.

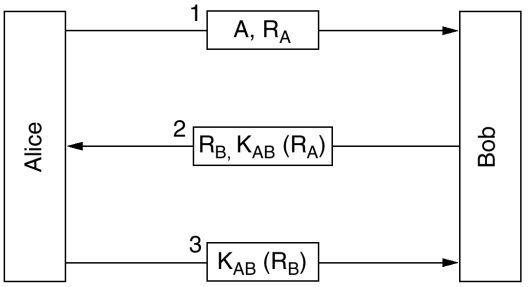


Fig. 8-33. Un protocol mai scurt de autentificare în doi pași.

Este acest nou protocol o îmbunătățire a celui original? Într-un anumit sens da: este mai scurt. Din nefericire, este și greșit. În anumite situații, Trudy poate învinge acest protocol folosind ceea ce se numește **atacul prin reflexie**. În particular, Trudy îl poate sparge dacă este posibil să deschidă sesiuni multiple cu Bob simultan. Această situație ar fi adevărată, de exemplu, dacă Bob este o bancă și este pregătit să accepte simultan mai multe conexiuni cu mașini de efectuat plăți.

Atacul prin reflexie al lui Trudy este prezentat în fig. 8-34. Pentru început, Trudy pretinde că este Alice și trimite R_T. Bob răspunde, ca de obicei, cu propria sa provocare, R_B. Acum Trudy este în impas. Ce poate să facă? Nu cunoaște $K_{AB}(R_B)$.

Ea poate deschide o a doua sesiune cu mesajul 3, furnizând R_B luat din mesajul 2 ca provocare a ei. Bob îl criptează calm și trimite înapoi $K_{AB}(R_B)$ în mesajul 4. Am colorat mesajele din sesiunea a doua pentru a le evidenția. Acum Trudy are informația care îi lipsea, așa că poate să finalizeze prima sesiune și să o abandoneze pe cea de-a doua. Bob este acum convins că Trudy este Alice, așa că atunci când ea întreabă de balanța contului său bancar, el i-o dă fără nici o problemă. Apoi când ea îi cere să transfere toți banii la o bancă secretă din Elveția, el face asta fără nici un moment de ezitare.

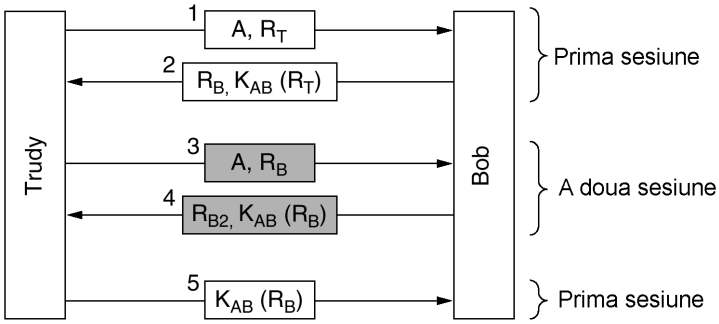


Fig. 8-34. Atacul prin reflexie.

Morala aceste povestiri este:

A proiecta un protocol corect de autentificare este mai greu decât pare.

Următoarele patru reguli generale sunt adeseori utile:

1. Inițiatorul să dovedească cine este înaintea celui care răspunde. În acest caz, Bob transmite informații importante înainte ca Trudy să-i fi dat vreo dovadă că este cine pretinde a fi.
2. Inițiatorul și cel ce răspunde să folosească chei diferite pentru dovadă, chiar dacă aceasta presupune existența a două chei partajate K_{AB} și K'_{AB}.
3. Inițiatorul și cel ce răspunde să-și extragă provocările din mulțimi diferite. De exemplu, inițiatorul trebuie să folosească numere pare, iar cel ce răspunde să folosească numere impare.
4. Protocolul să fie rezistent la atacuri ce implică o a doua sesiune paralelă, în care informația obținută într-o sesiune este folosită într-o alta.

Dacă fie și una singură din aceste patru reguli este nerespectată, deseori protocolul poate fi spart. În acest caz, patru reguli au fost încălcate cu consecințe dezastroase.

Acum să ne întoarcem și să ne uităm mai bine la protocolul din fig. 8-32. Sigur acel protocol nu poate fi ținta unui atac prin reflexie? Depinde. Este destul de subtil. Trudy a reușit să învingă protocolul nostru folosind un atac prin reflexie pentru că a fost posibil să deschidă o a doua sesiune cu Bob și să îl păcălească, făcându-l să răspundă la propriile întrebări. Ce s-ar întâmpla dacă Alice ar fi un calculator de uz general care ar accepta și el sesiuni multiple, și nu o persoană în fața unui calculator? Să vedem ce poate face Trudy.

Pentru a vedea cum funcționează atacul lui Trudy, a se vedea fig. 8-35. Alice începe prin a-și anunța identitatea în mesajul 1. Trudy interceptează acest mesaj și își începe propria sesiune cu mesajul 2, pretinzând că este Bob. Am colorat din nou mesajele din sesiunea a doua. Alice răspunde la mesajul 2 spunând: Pretinzi că ești Bob? Demonstrează. în mesajul 3. În acest moment Trudy e în impas pentru că nu poate demonstra că este Bob.

Ce face Trudy acum? Se întoarce la prima sesiune, în care e rândul ei să trimită o provocare, și trimite R_A pe care a primit-o în mesajul 3. Alice îi răspunde politicos în mesajul 5, furnizându-i lui Trudy informația de care are nevoie pentru a trimite mesajul 6 din sesiunea 2. În acest moment Trudy este practic liberă pentru că a răspuns cu succes la provocarea lui Alice în sesiunea 2. Acum poate să abandoneze sesiunea 1, să trimită un număr oarecare pentru restul sesiunii 2 și va avea o sesiune autentificată cu Alice în sesiunea 2.

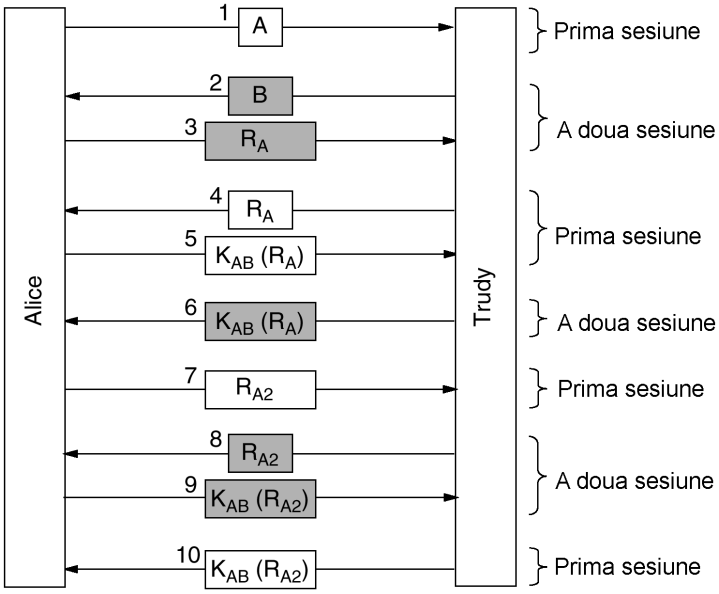


Fig. 8-35. Atac prin reflexie asupra protocolului din fig. 8-32.

Dar Trudy e periculoasă și chiar vrea să pună sare pe rană. În loc să trimită un număr oarecare pentru a completa sesiunea 2, ea așteaptă ca Alice să trimită mesajul 7, provocarea lui Alice pentru sesiunea 1. Bineînțeles, Trudy nu știe cum să răspundă, așa că folosește din nou atacul prin reflexie, trimițând R_{A2} ca mesajul 8. Alice criptează R_{A2} în mesajul 9. Acum Trudy se întoarce la sesiunea 1 și îi trimite lui Alice numărul dorit în mesajul 10, copiat din ceea ce trimisese Alice în mesajul 9. Acum Trudy are două sesiuni complet autentificate cu Alice.

Acest atac are un rezultat oarecum diferit de atacul protocolului cu trei mesaje reprezentat în fig. 8-34. De această dată, Trudy are două sesiuni autentificate cu Alice. În exemplul anterior, ea avea o singură sesiune autentificată cu Bob. Și aici, dacă am fi aplicat toate regulile generale pentru protocoale de autentificare discutate mai sus, acest atac ar fi putut fi oprit. O discuție detaliată despre acest gen de atacuri și despre cum să le împiedicăm se găsește în (Bird et. al, 1993). Ei arată și cum este posibilă construirea sistematică de protocoale pentru care să se poată demonstra că sunt corecte. Cel mai simplu protocol de acest fel este totuși cam complicat, așa că acum vom prezenta o altă clasă de protocoale care funcționează de asemenea corect.

Noul protocol de autentificare este reprezentat în fig. 8-36 (Bird ș. a., 1993). Acesta folosește un HMAC de tipul pe care l-am văzut când am studiat IPsec. Alice începe prin a-i trimite lui Bob un număr ad-hoc, R_A, ca mesajul 1. Bob răspunde alegând propriul număr ad-hoc, R_B, și trimițându-l înapoi împreună cu un HMAC. HMAC-ul este format prin construirea unei structuri de date alcătuită din numărul ad-hoc al lui Alice, numărul ad-hoc al lui Bob, identitățile lor și cheia secretă partajată, K_{AB}. Această structură de date este codificată prin dispersie în HMAC, de exemplu folosind SHA-1. Când Alice primește mesajul 2, ea are R_A (pe care a ales-o ea însăși), R_B, care ajunge ca text clar, cele două identități, și cheia secretă, K_{AB}, pe care a știut-o tot timpul, așa că poate să calculeze singură HMAC-ul. Dacă acesta coincide cu cel din mesaj, ea știe că vorbește cu Bob deoarece Trudy nu cunoaște K_{AB}, deci nu-și poate da seama ce HMAC să trimită. Alice îi răspunde lui Bob cu un HMAC ce conține doar cele două numere ad-hoc.

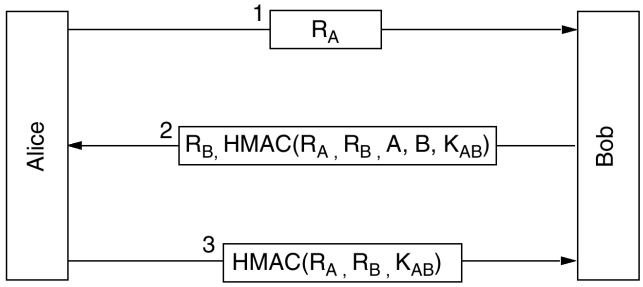


Fig. 8-36. Autentificarea folosind HMAC.

Poate Trudy să spargă în vreun fel acest protocol? Nu, pentru că nu poate forța nici una din părți să cripteze sau să rezume o valoare aleasă de ea, cum se întâmpla în fig. 8-34 și în fig. 8-35. Ambele HMAC-uri includ valori alese de partea care le trimite, ceea ce Trudy nu poate controla.

Utilizarea HMAC-urilor nu este singurul mod de a folosi această idee. O schemă alternativă deseori utilizată în locul calculării HMAC pentru o serie de elemente este criptarea acestor elemente secvențial, folosind înlănțuirea blocurilor cifrate.

8.7.2 Stabilirea unei chei secrete: schimbul de chei Diffie-Hellman

Până acum am presupus că Bob și Alice partajează o cheie secretă. Să presupunem că nu este așa (pentru că deocamdată nu există o PKI universal acceptată pentru semnarea și distribuirea certificatelor). Cum pot ei să stabilească una? O modalitate ar fi ca Alice să-l sune pe Bob și să-i dea cheia ei prin telefon, dar el probabil va începe întrebând: Cum știu eu că ești Alice și nu Trudy? Ei ar putea încerca să aranjeze o întâlnire la care fiecare din ei să aducă un pașaport, un permis de conducere și trei cărți de credit semnificative, dar fiind oameni ocupați, e posibil să nu poată găsi, vreme de luni de zile, o dată acceptabilă pentru amândoi. Din fericire, oricât de incredibil ar părea, există un mod ca cei ce sunt total străini să stabilească, chiar la lumina zilei, o cheie secretă partajată, cu Trudy înregistrând grijulie fiecare mesaj.

Protocolul care permite străinilor să stabilească o cheie secretă partajată se numește **interschimbul de chei Diffie-Hellman** (Diffie și Hellman, 1976) și funcționează după cum urmează. Alice și Bob trebuie să se pună de acord asupra a două numere mari, n și g, unde n este prim, $(n-1)/2$ este de asemenea prim și g îndeplinește anumite condiții. Aceste numere pot fi publice, astfel că oricare din ei poate pur și simplu să aleagă n și g și să o spună celuilalt, în mod deschis. Acum Alice alege un număr mare (să spunem pe 512 biți), x, și îl păstrează secret. Similar, Bob alege un număr mare, secret, y.

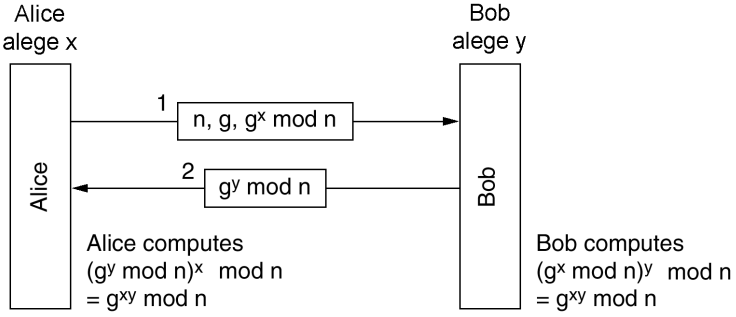


Fig. 8-37. Schimbul de cheie Diffie-Hellman.

Alice inițiază protocolul de schimb al cheii trimițându-i lui Bob un mesaj ce conține (n, g, g^x mod n), după cum se arată în fig. 8-37. Bob răspunde trimițându-i lui Alice un mesaj ce conține g^y mod n. Acum Alice are numărul pe care i l-a trimis Bob și îl ridică la puterea x modulo n pentru a obține (g^y mod n)^x mod n. Bob efectuează o operație similară pentru a obține (g^x mod n)^y mod n. Conform legilor aritmeticii modulare, ambele calcule duc la g^{xy} mod n. Iată cum Alice și Bob partajează acum o cheie secretă g^{xy} mod n.

Este evident că Trudy vede ambele mesaje. Ea cunoaște pe g și pe n din mesajul 1. Dacă ea ar putea calcula x și y ar putea să descopere cheia secretă. Necazul este că, dat fiind doar g^x mod n, ea nu poate afla pe x. Nu este cunoscut nici un algoritm practic pentru calculul logaritmilor discreți modulo un număr prim foarte mare.

Pentru a face exemplul anterior mai concret, vom folosi (complet nerealist) valorile $n = 47$ și $g = 3$. Alice alege $x = 8$ și Bob alege $y = 10$. Ambele chei sunt păstrate secrete. Mesajul lui Alice către Bob este (47, 3, 28) deoarece 3^8 mod 47 este 28. Mesajul lui Bob către Alice este (17). Alice calculează 17^8 mod 47, care este 4. Bob calculează 28^{10} mod 47, care este 4. Alice și Bob au determinat independent cheia secretă care este 4. Trudy are de rezolvat ecuația 3^x mod $47 = 28$, ceea ce poate fi făcut prin căutare exhaustivă în cazul unor numere mici ca acestea, dar nu și atunci când toate numerele sunt lungi de sute de biți. Toți algoritmii cunoscuți la ora actuală iau prea mult timp, chiar și atunci când sunt rulați folosind un supercalculator masiv paralel.

În ciuda eleganței algoritmului Diffie-Hellman, există o problemă: când Bob ia tripletul (47, 3, 28), cum știe el că este de la Alice și nu de la Trudy? Nu există nici o modalitate pentru aceasta. Din nefericire Trudy poate exploata acest fapt pentru a-i înșela atât pe Alice cât și pe Bob, după cum este ilustrat în fig. 8-38. Aici, când Alice și Bob îl aleg pe x, respectiv pe y, Trudy alege propriul său număr aleator, z. Alice trimite mesajul 1, destinat lui Bob. Trudy îl interceptează și trimite mesajul 2 lui Bob, folosind g și n corecți (care sunt disponibili public), dar cu al său z în loc de x. De asemenea ea trimite mesajul 3 lui Alice. Mai târziu Bob îi trimite lui Alice mesajul 4, pe care Trudy îl interceptează din nou și îl păstrează.

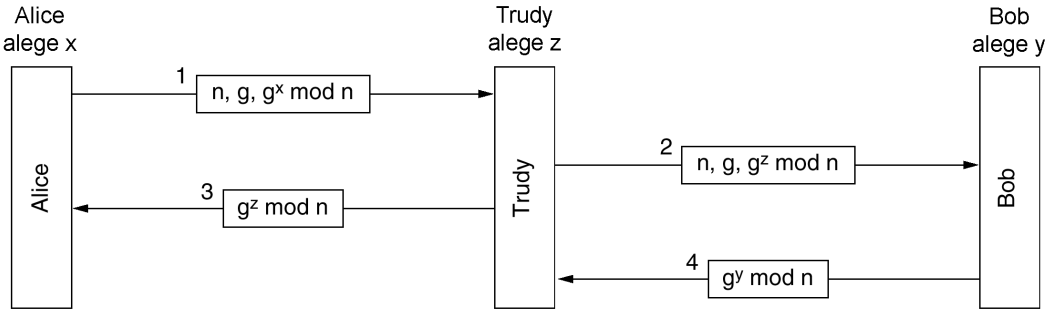


Fig. 8-38. Atacul de tip găleata brigăzii sau omul-din-mijloc.

Acum fiecare efectuează aritmetica modulară. Alice calculează cheia secretă ca fiind g^{xz} mod n și la fel face și Trudy (pentru mesajele lui Alice). Bob calculează g^{yz} mod n și la fel face și Trudy (pentru mesajele lui Bob). Alice crede că vorbește cu Bob, așa că ea stabilește o sesiune de cheie (cu Trudy). La fel face și Bob. Fiecare mesaj pe care Alice îl trimite în sesiunea criptată este capturat de Trudy, memorat, modificat la dorință și apoi (opțional) transmis lui Bob. Similar în cealaltă direcție. Trudy vede orice și poate modifica toate mesajele la dorință, în timp ce atât Alice cât și Bob trăiesc cu iluzia că au un canal de comunicație sigur de la unul la celălalt. Acest atac este cunoscut sub nu-

mele de **atacul găleata brigăzii de pompieri** (eng.: *bucket brigade attack*), deoarece el seamănă vag cu un departament de pompieri de pe vremuri trecând din mână în mână găleţile de-a lungul drumului de la maşina de pompieri la foc. El se mai numeşte şi **atacul omul-din-mijloc** (eng.: *man-in-the-middle attack*).

8.7.3 Autentificarea folosind un Centru de Distribuţia Cheilor

Stabilirea unui secret partajat cu un străin a mers destul de bine, dar nu în întregime. Pe de altă parte, probabil că nici nu merită să fie făcut (atacul strugurilor acri). Pentru a vorbi cu n oameni în acest mod ar fi necesare n chei. Pentru persoanele foarte cunoscute, gestiunea cheilor ar deveni o adevărată pacoste, în special dacă fiecare cheie trebuie stocată separat pe câte o cartelă de plastic.

O abordare diferită o reprezintă introducerea unui centru autorizat de distribuţie a cheilor (KDC - Key Distribution Center). În acest model, fiecare utilizator are o singură cheie partajată cu KDC. Autentificarea şi gestiunea cheilor de sesiune merg acum prin intermediul KDC. Cel mai simplu protocol cunoscut pentru autentificarea KDC, implicând două părţi şi un centru autorizat, este ilustrat în fig. 8-39.

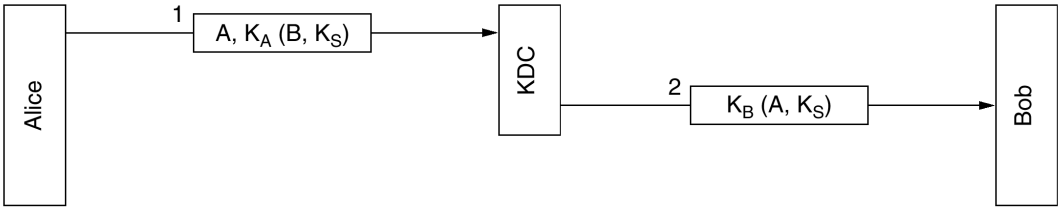


Fig. 8-39. O primă încercare de protocol de autentificare folosind un KDC .

Ideea din spatele protocolului este simplă: Alice alege o cheie de sesiune , K_S, şi anunţă KDC că vrea să vorbească cu Bob folosind K_S. Acest mesaj este criptat cu cheia secretă pe care Alice o împarte (numai) cu KDC, K_A. KDC decriptează acest mesaj, extrage identitatea lui Bob şi cheia de sesiune. Apoi el construieşte un nou mesaj ce conţine identitatea lui Alice şi cheia de sesiune şi trimite acest mesaj lui Bob. Criptarea este făcută cu K_B, cheia secretă pe care Bob o împarte cu KDC. Când Bob decriptează mesajul, el află că Alice vrea să vorbească cu el şi cheia pe care aceasta vrea să o utilizeze.

Autentificarea are loc gratuit. KDC ştie că mesajul 1 trebuie să fi venit de la Alice, deoarece nimeni altcineva nu poate să-l cripteze cu cheia secretă a Alicei. Similar, Bob ştie sigur că mesajul 2 vine de la KDC, în care el are încredere, deoarece nimeni altcineva nu mai cunoaşte cheia lui secretă.

Din nefericire, acest protocol prezintă un defect grav. Trudy are nevoie de ceva bani, aşa că ea imaginează un serviciu pe care îl poate executa pentru Alice, face o ofertă atractivă şi obţine postul. După ce îşi face treaba, Trudy cere politicos lui Alice să-i plătească transferându-i banii prin bancă. Aşa că Alice stabileşte o cheie de sesiune cu bancherul ei, Bob. Apoi ea îi trimite lui Bob un mesaj prin care cere ca banii respectivi să fie transferaţi în contul lui Trudy.

Între timp, Trudy se întoarce la vechile ei obiceiuri, furturile prin reţea. Ea copiază atât mesajul 2 din fig. 8-39, cât şi cererea de transferare a banilor care îl urmează. Mai târziu ea le trimite lui Bob. Bob le ia şi gândeşte: „Alice probabil că a angajat-o din nou pe Trudy. Cu siguranţă că ea lucrează bine.” Bob transferă din nou o cantitate de bani egală cu prima din contul lui Alice în al lui Trudy. La câtva timp după cea de-a 50-a pereche de mesaje pe care o primeşte, Bob aleargă afară din biroul

său pentru a o găsi pe Trudy și a-i oferi un împrumut mare astfel ca ea să-și poată extinde afacerea ce se dovedește a fi atât de plină de succes. Problema se numește **atacul prin reluare**.

Sunt câteva soluții posibile la atacul prin reluare. Prima este de a include în fiecare mesaj o amprentă de timp. Astfel, dacă cineva primește un mesaj expirat, îl ignoră. Necazul cu această abordare este că într-o rețea ceasurile nu sunt niciodată perfect sincronizate, astfel încât va exista un întreg interval de timp în care o amprentă de timp este validă. Trudy poate retrimite mesajul în acest interval de timp fără să fie prinsă.

Cea de-a doua soluție este să se pună, în fiecare mesaj, un număr ad-hoc. Fiecare parte trebuie să-și rememoreze toate numerele ad-hoc folosite anterior și să respingă orice mesaj ce conține un număr ad-hoc folosit deja. Dar numerele ad-hoc trebuie rememorate la nesfârșit, chiar și atunci când Trudy încearcă să retrimită un mesaj vechi de 5 ani. De asemenea, dacă o mașină cade și își pierde lista de numere ad-hoc, ea va fi din nou vulnerabilă la un atac prin reluare. Amprentele de timp și numerele ad-hoc pot fi combinate pentru a limita timpul în care acestea din urmă nu trebuie șterse, dar este evident că protocolul devine mult mai complicat.

O abordare și mai sofisticată a autentificării este folosirea unui protocol provocare-răspuns multicăi. Un exemplu binecunoscut de astfel de protocol este protocolul de **autentificare Needham-Schroeder** (Needham și Schroeder, 1978). O variantă a acestuia este prezentată în fig. 8-40.

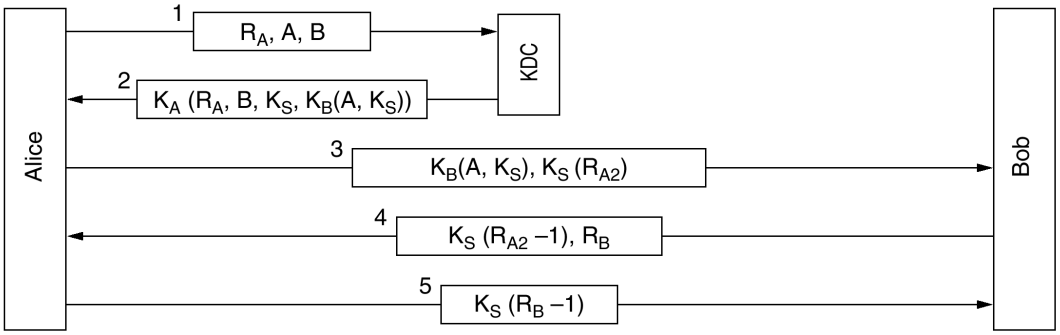


Fig. 8-40. Protocolul de autentificare Needham-Schroeder.

În acest protocol, Alice începe prin a anunța KDC că ea dorește să vorbească cu Bob. Acest mesaj conține un număr aleator mare, R_A, pe post de număr ad-hoc. KDC trimite înapoi mesajul 2 conținând numărul aleator al lui Alice împreună cu un tichet pe care ea îl poate trimite lui Bob. Scopul numărului aleator este acela de a o asigura pe Alice că mesajul 2 este proaspăt și nu unul reluat. Identitatea lui Bob este de asemenea inclusă pentru cazul în care lui Trudy îi vine amuzanta idee să înlocuiască *B*-ul din mesajul 1 cu propria sa identitate, astfel încât KDC să cripteze tichetul de la sfârșitul mesajului 2 cu K_T în loc de K_B. Tichetul criptat cu K_B este inclus în interiorul mesajului criptat, pentru a o împiedica pe Trudy să-l înlocuiască cu altceva pe drumul lui înapoi spre Alice.

Acum Alice îi trimite tichetul lui Bob, împreună cu un nou număr aleator, R_{A2}, criptat cu cheia de sesiune, K_S. În mesajul 4, Bob trimite înapoi $K_S(R_{A2}-1)$ pentru a-i dovedi lui Alice că vorbește cu adevăratul Bob. Trimiterea înapoi a lui $K_S(R_{A2})$ nu ar fi mers, deoarece ar fi fost posibil ca Trudy tocmai să-l fi furat din mesajul 3.

După primirea mesajului 4, Alice este convinsă că vorbește cu Bob și că până în acest moment nu s-au folosit mesaje reluate. Doar ea tocmai generase R_{A2} cu câteva milisecunde înainte. Scopul mesajului 5 este de a-l convinge pe Bob că cea cu care vorbește este chiar Alice și că nu s-au folosit nici

aici mesaje reluate. Posibilitatea oricărui tip de atac prin replicare este eliminată, deoarece fiecare parte nu numai că generează o provocare, dar și răspunde la una.

Cu toate că protocolul pare a fi destul de solid, el are o mică scăpare. Dacă Trudy reușește să obțină o cheie de sesiune veche în text clar, ea poate să inițieze o nouă sesiune cu Bob reluând mesajul 3 corespunzător cheii compromise și convingându-l pe acesta că ea este Alice (Denning și Sacco, 1981). De această dată ea poate prăda contul din bancă al lui Alice fără să trebuiască să se legitimeze nici măcar o dată.

Needham și Schroeder au publicat mai târziu un protocol care corectează această problemă (Needham și Schroeder, 1987). În același număr al aceluiași jurnal, Otway și Rees (1987) au publicat de asemenea un protocol care rezolvă problema pe o cale mai scurtă. Fig. 8-41 ilustrează un protocol Otway-Rees ușor modificat.

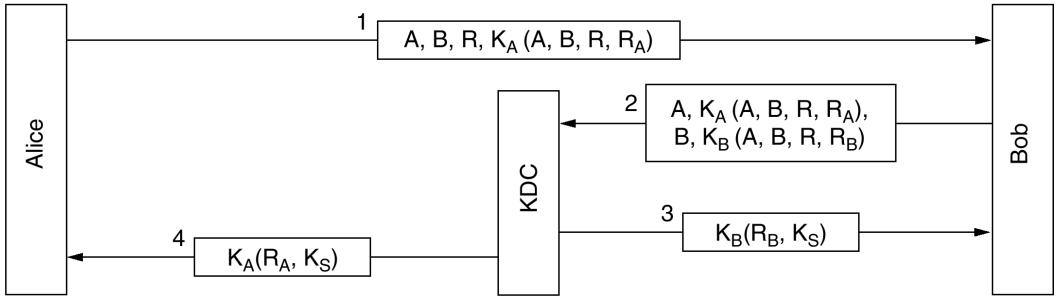


Fig. 8-41. Protocolul de autentificare Otway-Rees (puțin simplificat).

În protocolul Otway-Rees, Alice începe prin a genera o pereche de numere aleatoare, *R*, care va fi utilizată ca identificator comun, și R_A, pe care Alice îl va folosi pentru a-l provoca pe Bob. Când Bob preia acest mesaj, el construiește un mesaj nou din partea criptată a mesajului lui Alice și unul analog din partea sa. Ambele părți criptate cu K_A și K_B, identificând pe Alice și pe Bob, conțin identificatorul comun și o provocare.

KDC verifică dacă *R* din ambele părți este același. S-ar putea să nu fie, deoarece Trudy a intervenit cu *R* în mesajul 1 sau a înlocuit partea din mesajul 2. Dacă cele două R_S se potrivesc, KDC crede că mesajul de cerere de la Bob este valid. El generează atunci o cheie de sesiune și o criptează de două ori, o dată pentru Alice și o dată pentru Bob. Fiecare mesaj conține numărul aleator al receptorului, ca dovadă că mesajul a fost generat de KDC și nu de Trudy. În acest moment atât Alice cât și Bob sunt în posesia aceleiași chei de sesiune și pot începe comunicarea. Prima dată când ei vor schimba mesaje de date, fiecare va putea vedea că celălalt are o copie identică a lui K_S, astfel autentificarea fiind completă.

8.7.4 Autentificarea folosind Kerberos

Un protocol de autentificare folosit în multe sisteme reale (inclusiv Windows 2000) este **Kerberos**, care se bazează pe o variantă a protocolului Needham-Schroeder. Numele său vine de la un câine cu mai multe capete din mitologia greacă, ce era folosit pentru a păzi intrarea în Hades (probabil pentru a-i ține pe cei nedoriți afară). Kerberos a fost proiectat la M.I.T. pentru a permite utilizatorilor de la stațiile de lucru să acceseze resursele rețelei într-un mod sigur. Cea mai mare diferență dintre el și Needham-Schroeder este presupunerea lui că toate ceasurile sunt destul de bine sincronizate. Protocolul a trecut prin câteva iterații. V4 este versiunea cea mai larg utilizată în indus-

trie, așa că pe aceasta o vom descrie. După care vom spune câteva cuvinte despre succesoarea sa, V5. Pentru mai multe informații, vezi (Steiner et. al, 1988).

Kerberos implică trei servere în afară de Alice (stația de lucru a clientului):

Serverul de autentificare (AS – eng.: *Authentication Server*): verifică utilizatorii în timpul conectării.
Serverul de acordare a Tichetelor (TGS – eng.: *Ticket-Granting Server*): emite „demonstrarea identității tichetelor".
Serverul Bob: realizează efectiv acțiunea pe care o dorește Alice

AS este similar unui KDC prin aceea că el partajează o parolă secretă cu orice utilizator. Sarcina TGS este de a emite tichete care pot convinge serverele reale că acela care deține un tichet TGS este într-adevăr cel ce pretinde a fi.

Pentru a porni o sesiune, Alice stă așezată la o stație de lucru publică oarecare și-și tastează numele. Stația de lucru transmite numele ei la AS ca text clar, după cum este arătat în fig. 8-42. Ceea ce vine înapoi este o cheie de sesiune împreună cu un tichet $K_{TGS}(A, K_S)$, destinate TGS-ului. Aceste elemente sunt împachetate împreună și criptate folosind cheia secretă a lui Alice, astfel încât doar Alice să le poată decripta. Doar când sosește mesajul 2, stația de lucru îi va cere lui Alice parola. Parola este folosită pentru a genera K_A, în scopul decriptării mesajului 2 și obținerii cheii de sesiune și tichetului TGS din interiorul acestui mesaj. În acest moment, stația de lucru înlocuiește parola lui Alice pentru a se asigura că ea se găsește în interiorul stației de lucru pentru cel mult câteva milisecunde. Dacă Trudy încearcă să se conecteze ca Alice, parola pe care o introduce va fi greșită și stația de lucru va detecta acest lucru deoarece partea standard a mesajului 2 va fi incorectă.

După conectare Alice trebuie să anunțe stația de lucru că dorește să-l contacteze pe Bob, serverul de fișiere. Atunci stația de lucru trimite spre TGS mesajul 3 cerând un tichet pentru a-l folosi cu Bob. Elementul cheie în această cerere este $K_{TGS}(A, K_S)$, care este criptat cu cheia secretă a TGS și este folosit ca dovadă că transmițătorul este chiar Alice. TGS răspunde prin crearea unei chei de sesiune K_{AB}, pe care Alice să o folosească cu Bob. Se trimit înapoi două versiuni ale acesteia. Prima este criptată doar cu K_S, astfel încât Alice să poată să o citească. A doua este criptată cu cheia lui Bob, K_B, astfel încât Bob să o poată citi.

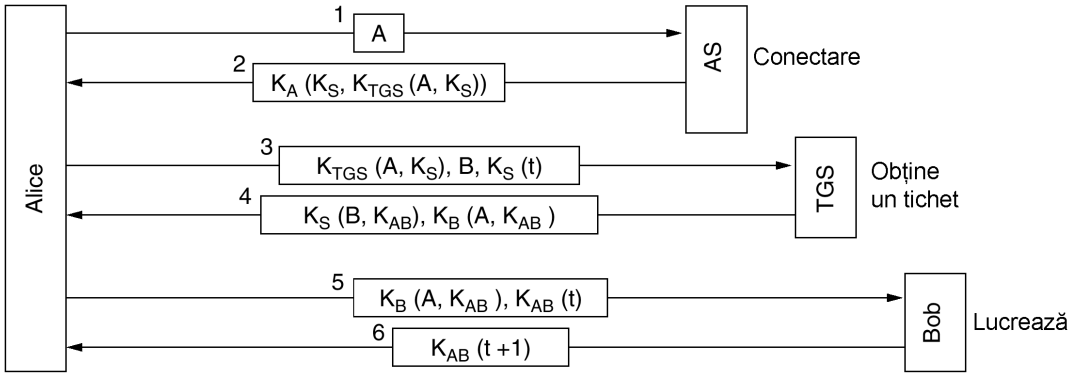


Fig. 8-42. Modul de operare la Kerberos V4.

Trudy poate copia mesajul 3 și poate încerca să-l utilizeze din nou, dar ea va fi împiedicată de amprenta de timp criptată, *t*, trimisă împreună cu el. Trudy nu poate înlocui amprenta de timp cu una mai recentă, deoarece ea nu cunoaște K_S, cheia de sesiune pe care o folosește Alice pentru a

vorbi cu TGS. Chiar dacă Trudy retrimite mesajul 3 repede, tot ceea ce va obține este o altă copie a mesajului 4, pe care nu a putut să-l decripteze prima dată și nu va putea nici a doua oară.

Acum Alice îi poate trimite lui Bob K_{AB} pentru a stabili o sesiune cu el. Acest schimb este de asemenea cu amprente de timp. Răspunsul este pentru Alice dovada că ea vorbește chiar cu Bob, nu cu Trudy.

După această serie de schimburi, Alice poate comunica cu Bob sub acoperirea cheii K_{AB}. Dacă mai târziu se decide că are nevoie să comunice cu alt server, Carol, ea va repeta doar mesajul 3 spre TGS, specificând de această dată C în loc de B. TGS va răspunde cu promptitudine cu un tichet criptat cu K_G pe care Alice îl poate transmite lui Carol și pe care Carol îl va accepta ca dovadă că el vine de la Alice.

Scopul întregii acțiuni este că acum Alice poate accesa serverele din toată rețeaua într-un mod sigur și că parola sa nu va fi niciodată transmisă prin rețea. De fapt parola trebuie să existe pe stația de lucru doar pentru câteva milisecunde. Cu toate acestea, trebuie remarcat că fiecare server face propria sa autorizare. Atunci când Alice îi prezintă tichetul său lui Bob, acest tichet doar îi demonstrează lui Bob cine l-a trimis. Ceea ce îi este permis lui Alice să facă, depinde doar de Bob.

Deoarece proiectanții Kerberos-ului nu s-au așteptat ca întreaga lume să aibă încredere într-un singur server de autentificare, ei au prevăzut posibilitatea de a avea **domenii** multiple, fiecare cu propriul său AS și TGS. Pentru a obține un tichet de la un server dintr-un domeniu de la distanță, Alice ar trebui să ceară propriului său TGS un tichet acceptat de TGS-ul din domeniul de la distanță. Dacă TGS-ul de la distanță este înregistrat la TGS-ul local (în același mod în care sunt serverele locale), TGS-ul local îi va da lui Alice un tichet valid pentru TGS-ul de la distanță. Astfel ea poate să lucreze acolo, de exemplu să ia tichete pentru serverele din acest domeniu. De notat totuși că, pentru ca părți din două domenii diferite să conlucreze, fiecare trebuie să se încreadă în TGS-ul celuilalt.

Kerberos V5 este mai sofisticat decât V4 și are o supraîncărcare mai mare. Pentru a descrie tipurile de date el folosește OSI ASN.1 (Abstract Syntax Notation 1) și prezintă mici modificări în protocoale. Mai mult decât atât, are timpi de viață mai mari pentru tichete, permite reînnoirea tichetelor și va elibera tichete postdatate. În plus, cel puțin în teorie, nu este dependent de DES, cum este V4, și suportă domenii multiple delegând servere de tichete multiple pentru generarea de tichete.

8.7.5 Autentificarea folosind criptografia cu cheie publică

Autentificarea mutuală poate fi realizată și cu ajutorul criptografiei cu cheie publică. Pentru început, Alice are nevoie de cheia publică a lui Bob. Dacă există o PKI cu un server de directoare care emite certificate pentru chei publice, Alice o poate cere pe a lui Bob, după cum se vede din fig. 8-43, ca mesajul 1. Răspunsul, în mesajul 2, este un certificat X.509 conținând cheia publică a lui Bob. Când Alice verifică faptul că semnătura e corectă, îi trimite lui Bob un mesaj conținând identitatea ei și un număr ad-hoc.

Când Bob primește acest mesaj, nu are idee dacă a venit de la Alice sau de la Trudy, dar continuă și cere serverului de directoare cheia publică a lui Alice (mesajul 4), pe care o obține imediat (mesajul 5). Apoi îi trimite lui Alice un mesaj conținând R_A a lui Alice, propriul număr ad-hoc, R_B, și o cheie de sesiune propusă, K_S, ca mesajul 6.

Când Alice primește mesajul 6, ea îl decriptează folosind propria cheie privată. Ea vede R_A în mesaj, ceea ce îi dă o senzație plăcută. Mesajul trebuie să fi venit de la Bob, pentru că Trudy nu are nici un mijloc de a determina R_A. Mai mult, trebuie să fie un mesaj proaspăt, și nu unul reluat, din moment ce ea tocmai i-a trimis R_A lui Bob. Alice acceptă cheia de sesiune trimițând înapoi mesajul 7. Când Bob vede R_B criptat cu cheia de sesiune pe care a generat-o el, știe că Alice a primit mesajul 6 și a verificat R_A.

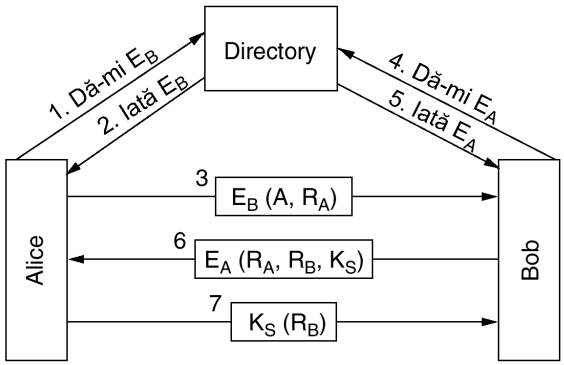


Fig. 8-43. Autentificarea mutuală folosind criptografia cu cheie publică.

Cum poate încerca Trudy să compromită acest protocol? Ea poate fabrica mesajul 3 și-l poate păcăli pe Bob să o testeze pe Alice, dar Alice va vedea un R_A pe care nu l-a trimis ea și nu va merge niciodată mai departe. Trudy nu poate falsifica mesajul 7 pentru Bob, deoarece nu știe R_B sau K_S și nu le poate determina fără cheia privată a lui Alice. De data aceasta ea nu are noroc.

8.8 CONFIDENȚIALITATEA POȘTEI ELECTRONICE

Când un mesaj de poștă electronică este trimis între două locații situate la distanță, acesta va trece în drumul său pe la un număr mare de mașini. Oricare dintre acestea poate să citească și să înregistreze mesajul pentru utilizare ulterioară. Confidențialitatea este inexistentă, în ciuda a ceea ce mulți oameni cred (Weisband și Reining, 1995). Cu toate acestea, multora le-ar plăcea să poată trimite mesaje care să fie citite doar de cei cărora le sunt adresate și de nimeni altcineva: nu de șeful lor și nici chiar de guvern. Această dorință a stimulat mai multe persoane și grupuri să aplice principiile criptografiei, studiate de noi anterior, la poșta electronică, pentru a produce un sistem de e-mail sigur. În secțiunile următoare vom studia un sistem de e-mail sigur larg utilizat, PGP, și apoi vom menționa pe scurt altele două, PEM și S/MIME. Pentru informații suplimentare a se vedea (Kaufman et. al, 2002; și Schneier, 1995).

8.8.1 PGP-Pretty Good Privacy (rom.: Confidențialitate Destul de Bună)

Primul nostru exemplu, **PGP** (eng.: **Pretty Good Privacy**) este în cea mai mare parte rodul gândirii unei singure persoane, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann este un avocat al confidențialității, al cărui motto este: Dacă dreptul la confidențialitate este în afara legii, atunci doar cei aflați în afara legii vor avea confidențialitate. Lansat în 1991, PGP este un pachet complet de securitate a e-mail-ului, care oferă confidențialitate, autentificare, semnături digitale și compresie, toate într-o formă ușor de utilizat. Pe deasupra, pachetul complet, incluzând tot codul

sursă, este distribuit gratuit prin Internet. Datorită calității sale, prețului (zero) și disponibilității pe platformele UNIX, Linux, Windows și Mac OS, acesta este larg utilizat astăzi.

PGP criptează datele folosind un cifru bloc numit **IDEA** (eng.: **International Data Encryption Algorithm**, rom.: **Algoritm Internațional de Criptare a Datelor**), care folosește chei de 128 de biți. Acesta a fost proiectat în Elveția într-un moment in care DES era considerat compromis și AES încă nu fusese inventat. Conceptual, IDEA este similar cu DES și AES: permută biții într-un șir de runde, dar detaliile funcțiilor de permutare sunt diferite de DES și AES. Gestiunea cheilor folosește RSA și integritatea datelor folosește MD5, subiecte pe care le-am discutat deja.

PGP a fost implicat în diferite controverse începând din prima zi (Levy, 1993). Deoarece Zimmermann nu a făcut nimic pentru a împiedica alte persoane să distribuie PGP pe Internet, de unde poate fi luat de oameni din toată lumea, guvernul Statelor Unite a pretins că Zimmermann a violat legile privind exportul de muniție. Investigarea lui Zimmermann de către guvernul Statelor Unite a durat 5 ani, dar până la urmă a încetat, probabil din două motive. În primul rând, Zimmermann nu a pus el însuși PGP pe Internet, așa că avocatul lui a susținut că *el* nu a exportat niciodată nimic (și apoi mai este și problema dacă de fapt a crea un sit Web constituie un export). În al doilea rând, guvernul și-a dat seama în cele din urmă că a câștiga un proces înseamnă a convinge jurații că un sit Web conținând un program de confidențialitate ce poate fi descărcat intră sub incidența legii traficului de arme, care interzice exportul de material de război cum ar fi tancurile, submarinele, aparatele de zbor militare și armele nucleare. Și probabil că nici anii întregi de publicitate negativă nu au ajutat prea mult.

Făcând o paranteză, legile exportului sunt bizare, ca să folosim termeni moderați. Guvernul a considerat punerea de cod pe un sit Web ilegală și l-a hărțuit pe Zimmermann 5 ani pentru acest lucru. Pe de altă parte, când cineva a publicat codul sursă complet al PGP, în C, într-o carte (cu caractere mari și cu sume de control pe fiecare pagină, ca să poată fi verificat ușor) și apoi a exportat cartea, nu au fost probleme cu guvernul deoarece cărțile nu sunt clasificate ca muniții. Sabia este mai puternică decât condeiul, cel puțin pentru Unchiul Sam.

O altă problemă cu care s-a confruntat PGP a fost una de încălcare a patentului. Compania ce deține patentul RSA, RSA Security, Inc., a susținut că folosirea de către PGP a algoritmului RSA îi încalcă patentul, dar această problemă a fost rezolvată în versiunile începând de la 2.6. Mai mult, PGP folosește și alt algoritm de criptare patentat, IDEA, ceea ce a provocat niște probleme la început.

Cum sursele PGP sunt publice, diverse persoane și grupuri l-au modificat și au produs un număr de versiuni. Unele au fost proiectate ca să ocolească problemele cu legile muniției, altele s-au orientat spre evitarea folosirii algoritmilor patentați, și altele au vrut să-l transforme într-un produs comercial, în care sursele să nu mai fie disponibile. Deși legile muniției au mai fost liberalizate puțin (altfel produsele ce folosesc AES nu ar fi putut fi exportate din S.U.A.) și patentul RSA a expirat în septembrie 2000, consecința tuturor acestor probleme este că acum sunt în circulație diverse versiuni incompatibile ale PGP, sub nume diferite. Discuția de mai jos se referă la PGP clasic, adică cea mai veche și mai simplă versiune. Altă versiune populară, Open PGP, este descrisă în RFC 2440. Încă o altă versiune este GNU Privacy Guard.

În mod intenționat PGP utilizează algoritmi de criptare existenți, în loc să inventeze unii noi. Se bazează în principal pe algoritmi care au fost supuși unor numeroase analize amănunțite și nu au fost proiectați sau influențați de vreo agenție guvernamentală care să încerce să le micșoreze puterea. Pentru cei ce au tendința să nu aibă încredere în guvern, această proprietate este un mare avantaj.

PGP permite compresia de text, asigurarea secretului mesajelor și semnături digitale și furnizează de asemenea facilități de management extensiv al cheilor, dar, destul de ciudat, nu și facilități pen-

tru e-mail. Este mai mult un preprocesor care preia text clar de la intrare și produce text semnat, cifrat, în bază 64, ca rezultat. Acest rezultat poate fi apoi trimis prin e-mail, bineînțeles. Unele implementări ale PGP apelează, ca pas final, un agent utilizator care să trimită efectiv mesajul.

Pentru a vedea cum funcționează PGP, să considerăm exemplul din fig. 8-44. Aici, Alice vrea să-i trimită lui Bob, într-o manieră sigură, un mesaj text simplu, semnat, P. Alice și Bob au cheile private *(D_X)* și cheile RSA publice *(E_X)*. Să presupunem că fiecare știe cheia publică a celuilalt; ne vom ocupa mai târziu de administrarea cheilor.

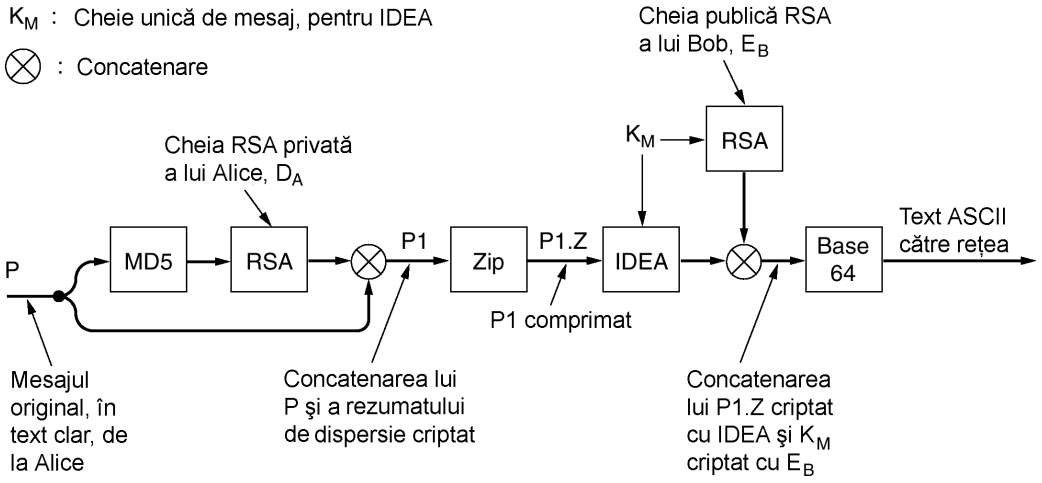


Fig. 8-44. PGP în acțiunea de trimitere a unui mesaj.

Alice începe prin a invoca programul PGP pe calculatorul său. Mai întâi PGP rezumă prin dispersare (eng.: hash) mesajul P utilizând MD5, apoi criptează codul de dispersie rezultat utilizând cheia sa RSA privată, D_A. Când, în cele din urmă, Bob primește mesajul, el poate decripta rezumatul cu cheia publică cunoscută a lui Alice și poate testa corectitudinea acestuia. Chiar dacă altcineva (de ex. Trudy) poate obține codul în această etapă și îl poate decripta cu cheia publică a lui Alice, puterea lui MD5 garantează că este nerealizabilă computațional producerea unui alt mesaj care să aibă același cod MD5.

Rezumatul criptat și mesajul original sunt acum concatenate într-un singur mesaj *P1* și comprimate apoi cu programul ZIP, care utilizează algoritmul Ziv-Lempel (Ziv și Lempel, 1977). Numim ieșirea obținută la acest pas *P1.Z*.

Apoi, PGP îi cere lui Alice introducerea unui șir de caractere oarecare. Atât conținutul acestuia cât și viteza de tastare sunt utilizate pentru a genera o cheie de mesaj de tip IDEA, de 128 de biți, K_M (numită cheie de sesiune în literatura PGP, dar numele este nepotrivit atâta timp cât nu există nici o sesiune). K_M este acum utilizat pentru a cripta *P1.Z* cu IDEA, prin metoda de tip reacție cifrată. În plus, K_M este criptată cu cheia publică a lui Bob, E_B. Aceste două componente sunt apoi concatenate și convertite în bază 64, așa cum s-a discutat în secțiunea despre MIME din Cap. 7. Mesajul rezultat conține numai litere, cifre și simbolurile +, / și =, ceea ce înseamnă că poate fi pus într-un corp de RFC 822 și că ne putem aștepta să ajungă nemodificat la destinație.

Când Bob primește mesajul, îl reconvertește din bază 64 și decriptează cheia IDEA utilizând cheia sa RSA privată. Utilizând această cheie, decriptează mesajul pentru a obține *P1.Z*. După de-

compresia acestuia, Bob separă textul simplu de codul cifrat și decriptează codul de dispersie utilizând cheia publică a lui Alice. Dacă codul textului clar coincide cu ceea ce a calculat el utilizând MD5, el știe că P este mesajul corect și că provine de la Alice.

Merită observat că RSA este utilizat doar în două locuri aici: pentru a cifra codul de dispersie de 128 de biți generat de MD5 și pentru a cifra cheia IDEA de 128 de biți. Deși RSA este lent, are de criptat doar 256 de biți și nu un volum mare de date. Mai mult, toți cei 256 de biți de text simplu sunt generați extrem de aleator, astfel încât numai pentru a determina dacă o cheie ghicită este corectă, Trudy ar trebui să depună o cantitate însemnată de muncă. Criptarea de mare putere este realizată de IDEA, care este cu câteva ordine de mărime mai rapidă decât RSA. Astfel, PGP asigură securitate, compresie și o semnătură digitală și face acest lucru într-o manieră chiar mult mai eficientă decât schema ilustrată în fig. 8-19.

PGP acceptă trei lungimi de chei RSA. Rămâne la latitudinea utilizatorului să o aleagă pe cea mai potrivită. Lungimile disponibile sunt:

1. Obișnuită (384 biți): poate fi spartă ușor în ziua de azi.
2. Comercială (512 biți): ar putea fi spartă de organizații cu nume din trei litere (care se ocupă cu securitatea statului).
3. Militară (1024 biți): Nu poate fi spartă de nici un pământean.
4. Extraterestră (2048 biți): Nu poate fi spartă nici de cineva de pe altă planetă.

Cum RSA este folosită doar pentru două calcule ușoare, toată lumea ar trebui să folosească mereu chei de lungime extraterestră.

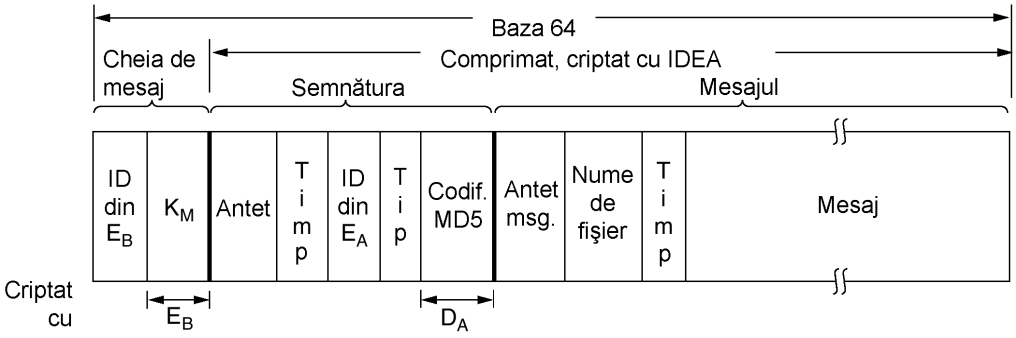


Fig. 8-45. Un mesaj PGP

Formatul unui mesaj PGP este prezentat în fig. 8-45. Mesajul are trei părți, conținând cheia IDEA, semnătura și respectiv mesajul. Partea care conține cheia mai include, de asemenea, un identificator de cheie, deoarece utilizatorilor li se permite să aibă mai multe chei publice.

Partea cu semnătura conține un antet, de care nu ne vom ocupa aici. Antetul este urmat de: o amprentă de timp, identificatorul pentru cheia publică a emițătorului, care poate fi folosit pentru a decripta codul semnăturii, unele informații care identifică algoritmul folosit (pentru a permite folosirea algoritmilor MD6 și RSA2 atunci când aceștia vor fi inventați) precum și codul de dispersie criptat.

Partea de mesaj conține de asemenea un antet, numele implicit care va fi folosit pentru fișier în cazul în care utilizatorul dorește să-l salveze pe disc, o amprentă de timp pusă la crearea mesajului și, în sfârșit, mesajul în sine.

Administrarea cheilor a fost tratată cu atenție foarte mare în PGP, deoarece acesta este călcâiul lui Ahile pentru sistemele de securitate. Administrarea cheilor funcționează după cum urmează. Fiecare utilizator menține două structuri de date locale: un inel cu chei private și unul cu chei publice. **Inelul cheilor private** conține una sau mai multe perechi de chei personale (privată-publică). Motivul pentru care se acceptă mai multe chei pentru un utilizator este pentru a permite acestora să-și schimbe cheile publice periodic sau când se consideră că acestea au fost compromise, fără a invalida mesajele aflate în pregătire sau în tranzit. Fiecare pereche are un identificator asociat, astfel încât un emițător de mesaj poate spune destinatarului ce cheie publică a fost folosită pentru criptarea acestuia. Identificatorii de mesaj constau din ultimii 64 de biți ai cheii publice. Utilizatorii sunt responsabili pentru evitarea conflictelor între identificatorii cheilor lor publice. Cheile private de pe disc sunt criptate folosind o parolă specială (arbitrar de lungă) pentru a le proteja împotriva unor atacuri.

Inelul cheilor publice conține cheile publice ale corespondenților utilizatorului. Acestea sunt necesare pentru criptarea cheilor de mesaj asociate cu fiecare mesaj. Fiecare intrare în inelul cheilor publice conține nu doar cheia publică, ci și identificatorul său pe 64 de biți și o indicație asupra încrederii pe care o are utilizatorul în cheie.

Problema care trebuie rezolvată aici este următoarea. Să presupunem că cheile publice sunt menținute în grupurile de știri. O metodă pentru Trudy de a citi e-mail-ul secret al lui Bob este de a ataca grupul de știri și de a înlocui cheia publică a lui Bob cu una la alegerea sa. Când Alice va lua mai târziu așa-zisa cheie a lui Bob, Trudy poate lansa un atac găleata-brigăzii la adresa lui Bob.

Pentru a preveni astfel de atacuri, sau cel puțin pentru a minimiza consecințele lor, Alice trebuie să știe cât de mult se poate încrede în obiectul numit „cheia lui Bob" din inelul său de chei publice. Dacă ea știe că Bob personal i-a dat o dischetă conținând cheia, atunci poate acorda valoarea maximă de încredere. Această abordare descentralizată, bazată pe controlul utilizatorului, asupra administrării cheilor publice este ceea ce face PGP să se deosebească de schemele centralizate PKI.

Totuși, în practică, oamenii primesc cheile publice întrebând un server de chei sigur. Din acest motiv, după ce X.509 a fost standardizat, PGP a început să suporte aceste certificate, pe lângă tradiționalul mecanism al inelelor cu chei publice. Toate versiunile curente de PGP suportă X.509.

8.8.2 PEM-Privacy Enhanced Mail (Poștă cu Confidențialitate Sporită)

Spre deosebire de PGP, care a fost inițial opera unui singur om, cel de-al doilea exemplu al nostru, **PEM** (eng.: **Privacy Enhanced Mail**), dezvoltat la sfârșitul anilor '80, este un standard oficial Internet și este descris în patru RFC-uri: RFC 1421 până la RFC 1424. Foarte pe scurt, PEM acoperă același teritoriu ca și PGP: confidențialitatea și autentificarea sistemelor de e-mail bazate pe RFC 822. Cu toate acestea, el prezintă și unele diferențe față de abordarea și tehnologia PGP.

Mesajele trimise folosind PEM sunt mai întâi convertite într-o formă canonică, astfel încât ele au aceleași convenții referitoare la spații albe (de ex. tab-uri, spațiile de la sfârșit de text). Apoi este calculat un cod de dispersie al mesajului, folosind MD2 sau MD5. După aceasta, concatenarea codului de dispersie și a mesajului este criptată folosind DES. În lumina cunoscutei slăbiciuni a unei chei de 56 de biți, această alegere este în mod sigur suspectă. Mesajul criptat poate fi apoi codificat utilizând o codificare în baza 64 și transmis destinatarului.

Ca și în PGP, fiecare mesaj este criptat cu o cheie unică, inclusă și ea în mesaj. Cheia poate fi protejată fie cu RSA, fie cu DES triplu folosind EDE.

În PEM administrarea cheilor este mult mai structurată decât în PGP. Cheile sunt certificate prin certificate X.509 emise de CA-uri, care sunt organizate într-o ierarhie rigidă pornind de la o

singură rădăcină (bază). Avantajul acestei scheme este că revocarea certificatului este posibilă dacă autoritatea corespunzătoare rădăcinii emite periodic CRL-uri.

Singura problemă cu PEM este că nimeni nu l-a folosit niciodată și a dispărut de mult în neant. În mare, problema a fost politică: cine ar funcționa ca bază a ierarhiei și în ce condiții? Nu a fost lipsă de candidați, dar multora le-a fost frică să încredințeze vreuneia dintre companii securitatea întregului sistem. Cel mai serios candidat, RSA Security, Inc., a vrut să perceapă o taxă pentru fiecare certificat emis. Totuși, unele organizații au respins ideea. În particular, guvernul Statelor Unite poate folosi toate patentele din țară fără a plăti, iar companiile din afara S.U.A. se obișnuiseră să folosească algoritmul RSA pe gratis (compania a uitat să îl patenteze în afara Statelor Unite). Nici unii, nici alții nu erau entuziasmați de ideea de a trebui deodată să plătească RSA Security, Inc. pentru lucruri pe care ei le făcuseră dintotdeauna pe gratis. Până la urmă, nu a putut fi găsită o bază pentru ierarhie și PEM s-a prăbușit.

8.8.3 S/MIME

Următoarea aventură a IETF în domeniul securității poștei electronice, denumită **S/MIME** (**Secure/MIME,** rom.: **MIME Sigur**), este descrisă în RFC 2632 până la 2643. Ca și PEM, acesta oferă autentificare, integritatea datelor, confidențialitate și non-repudiere. Este de asemenea destul de flexibil, suportând o varietate de algoritmi criptografici. În mod previzibil, având în vedere numele, S/MIME se integrează bine cu MIME, permițând tuturor tipurilor de mesaje să fie protejate. Sunt definite o varietate de antete noi MIME, de exemplu pentru a conține semnături digitale.

IETF a învățat evident ceva din experiența cu PEM. S/MIME nu are o ierarhie rigidă pentru certificate, pornind de la o singură bază. În schimb, utilizatorii pot avea mai multe puncte de încredere (eng.: trust anchors). Atâta timp cât un certificat poate fi urmărit înapoi până la un punct în care utilizatorul are încredere, este considerat valid. S/MIME folosește algoritmii și protocoalele standard pe care le-am examinat până acum, deci nu îl vom mai discuta aici. Pentru detalii, vă rog să consultați RFC-urile.

8.9 SECURITATEA WEB-ULUI

Am studiat două domenii importante în care securitatea este necesară: comunicațiile și poșta electronică. Puteți să vi le imaginați pe acestea ca fiind supa și aperitivul. Acum este momentul pentru felul principal: securitatea Web-ului. Web-ul este locul unde în zilele noastre cele mai multe Trudy își petrec vremea făcând răutăți. În secțiunile următoare vom analiza unele probleme și aspecte referitoare la securitatea Web-ului.

În linii mari, securitatea Web-ului poate fi împărțită în trei subiecte. În primul rând, cum pot fi sigur denumite obiectele și resursele? În al doilea rând, cum se pot stabili conexiuni sigure, autentificate? În al treilea rând, ce se întâmplă când un sit Web trimite unui client un fragment de cod executabil? După ce prezentăm câteva tipuri de pericole, vom examina toate aceste probleme.

8.9.1 Pericole

În ziare se poate citi despre problemele de securitate a siturilor Web aproape în fiecare săptămână. Situația e într-adevăr destul de proastă. Să vedem câteva exemple de lucruri care s-au întâmplat deja. În primul rând, paginile principale ale multor organizații au fost atacate și înlocuite cu alte pagini alese de spărgători (eng.: crackers). (Presa populară denumește persoanele care sparg sistemele „hackers", dar mulți programatori rezervă acest termen pentru marii programatori. Preferăm să denumim aceste persoane „spărgători" - „crackers"). Printre siturile care au fost sparte se numără și Yahoo, Armata S.U.A., CIA, NASA, New York Times. În cele mai multe cazuri, spărgătorii au pus acolo doar un text amuzant și siturile au fost reparate în câteva ore.

Acum să vedem niște cazuri mult mai serioase. Multe situri au căzut din cauza atacurilor de refuz al serviciilor (eng.: denial-of-service attacks), în care spărgătorul „inundă" situl cu trafic, făcându-l incapabil de a răspunde la cererile legitime. De multe ori atacul e lansat de pe un număr mare de mașini în care spărgătorul tocmai a intrat (atacuri distribuite de negare a serviciilor – DDoS attacks). Aceste atacuri au devenit atât de obișnuite încât acum nici nu mai ajung la știri, dar pot costa situl atacat mii de dolari în afaceri pierdute.

În 1999, un spărgător suedez a intrat în situl Hotmail al firmei Microsoft și a creat un sit oglindă (mirror) care permitea oricui să scrie numele unui utilizator Hotmail și apoi să citească toată poșta electronică, cea curentă și cea arhivată, a persoanei respective.

În alt caz, un spărgător rus de 19 ani numit Maxim a intrat într-un sit de comerț electronic și a furat 300000 de numere de cărți de credit. Apoi i-a abordat pe deținătorii sitului și le-a spus că dacă nu îl plătesc cu 100000$, va publica toate numerele de cărți de credit pe Internet. Ei nu au cedat șantajului și el chiar a publicat numerele cărților de credit, producând mari pagube multor victime inocente.

În alt stil, un student de 23 de ani din California a trimis prin e-mail unei agenții de presă un comunicat conținând afirmația falsă că Emulex, o corporație americană, va suferi mari pierderi și că directorul executiv își va da imediat demisia. În câteva ore, acțiunile companiei au scăzut cu 60%, provocând celor care le dețineau pierderi de peste 2 miliarde $. Autorul faptei a câștigat un sfert de milion de dolari vânzând acțiunile cu puțin timp înainte de a trimite anunțul. Chiar dacă acest eveniment nu a fost o spargere a unui sit Web, este clar că punerea unui astfel de anunț pe pagina unei mari corporații ar avea un efect similar.

Am putea (din nefericire) să continuăm așa pe multe pagini. Dar acum este timpul să examinăm unele aspecte tehnice legate de securitatea Web-ului. Pentru mai multe informații despre probleme de securitate de orice fel, vezi (Anderson, 2001; Garfinkel și Spafford, 2002; și Schneier, 2000). Și căutând pe Internet se va obține un număr imens de astfel de cazuri.

8.9.2 Siguranța numelor

Să începem cu ceva simplu: Alice vrea să viziteze situl Web al lui Bob. Ea tastează URL-ul lui Bob în browser și după câteva secunde, apare o pagină Web. Dar este pagina lui Bob? Pate că da, poate că nu. Poate că Trudy s-a întors din nou la farsele ei. De exemplu, ar putea să intercepteze toate pachetele ce vin de la Alice și să le examineze. Când găsește o cerere HTTP de tip *GET* pentru situl lui Bob, ar putea să se ducă ea însăși la situl lui Bob ca să ia pagina, să o modifice după cum dorește și să îi trimită lui Alice pagina falsă. Alice nu ar fi deloc mai înțeleaptă. Mai rău, Trudy ar putea să micșoreze prețurile din magazinul virtual al lui Bob pentru a face produsele să pară foarte atractive, păcălind-o astfel pe Alice să trimită numărul cărții ei de credit lui „Bob" ca să cumpere ceva.

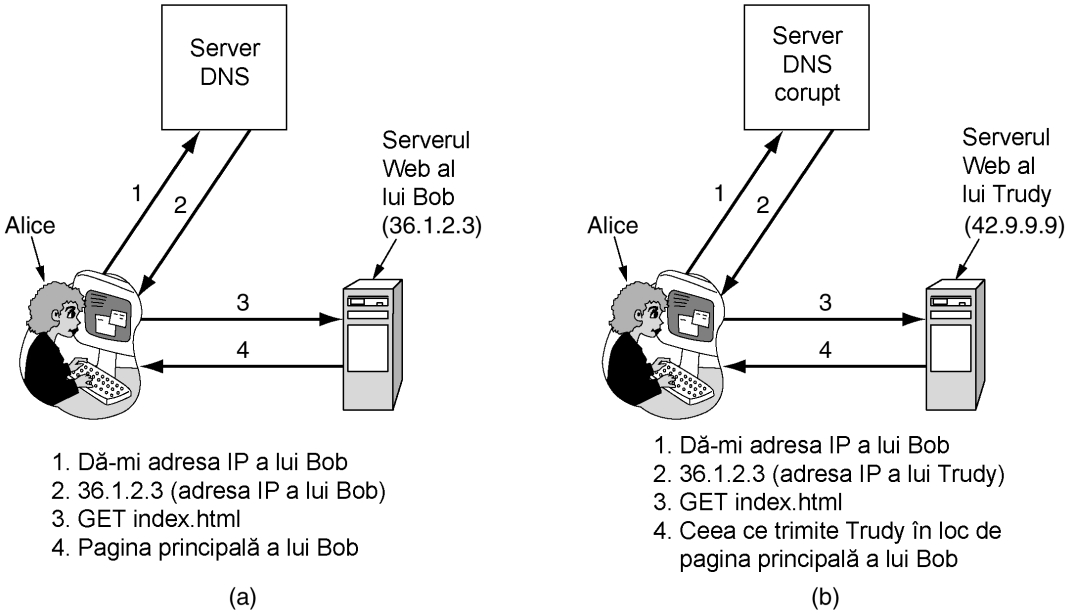


Fig. 8-46. (a) Situație normală. (b) Atac bazat pe spargerea DNS-ului și modificarea înregistrării lui Bob.

Un dezavantaj al acestui atac clasic de tip omul-din-mijloc este acela că Trudy trebuie să poată intercepta traficul ce pleacă de la Alice și să îl poată falsifica pe cel ce vine. Practic, ea trebuie să intercepteze linia de telefon a lui Alice sau a lui Bob, pentru că interceptarea fibrei optice este foarte dificilă. Deși interceptarea activă a cablurilor este cu siguranță posibilă, ea presupune o anumită cantitate de muncă, iar Trudy, cu toate că este inteligentă, este leneșă. În plus, există moduri mai ușoare de a o păcăli pe Alice.

Păcălirea DNS-ului (DNS Spoofing)

De exemplu, să presupunem că Trudy poate să spargă sistemul DNS, sau poate memoria ascunsă a DNS de la ISP-ul lui Alice, și să înlocuiască adresa IP a lui Bob (să zicem, 36.1.2.3) cu adresa ei (a lui Trudy, să zicem, 42.9.9.9). Aceasta duce la următorul atac. Modul în care ar trebui să funcționeze este ilustrat în fig. 8-46(a). Aici (1) Alice cere DNS-ului adresa IP a lui Bob, (2) o obține, (3) îi cere lui Bob pagina sa principală și (4) o obține și pe aceasta. După ce Trudy a modificat înregistrarea DNS corespunzătoare lui Bob astfel încât aceasta să conțină adresa ei IP în loc de cea a lui Bob, ajungem la situația din fig. 8-46(b). Aici, când Alice caută adresa IP a lui Bob, o obține pe a lui Trudy, așa că tot traficul ei destinat lui Bob ajunge la Trudy. Trudy poate organiza acum un atac omul-din-mijloc fără să mai fie necesar să facă efortul de a intercepta vreo linie telefonică. În schimb, ea trebuie să pătrundă într-un server DNS și să schimbe o înregistrare, ceea ce este mult mai ușor.

Cum ar putea Trudy să înșele DNS-ul? Acest lucru se dovedește a fi relativ ușor. Pe scurt, Trudy poate să păcălească serverul DNS de la ISP-ul lui Alice, făcându-l să trimită o cerere pentru a afla adresa lui Bob. Din nefericire, deoarece DNS-ul folosește UDP, serverul DNS practic nu poate verifica cine a dat cu adevărat răspunsul. Trudy poate profita de această proprietate pentru a falsifica răspunsul așteptat, introducând astfel o adresă IP falsă în memoria ascunsă a serverului DNS.

Pentru simplitate, vom presupune că inițial ISP-ul lui Alice nu are o intrare pentru situl Web al lui Bob, *bob.com*. Dacă are, Trudy poate să aștepte până când aceasta expiră și să încerce mai târziu (sau să folosească alte manevre).

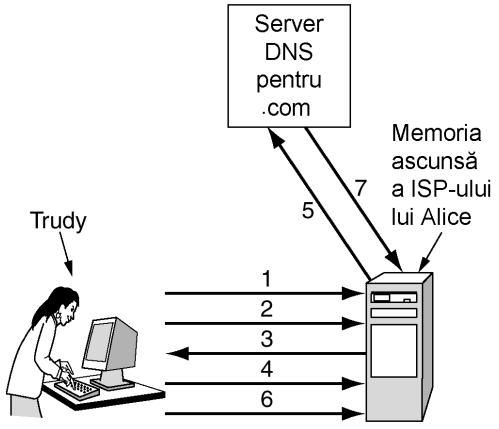
Trudy începe atacul prin a trimite o cerere de căutare ISP-ului lui Alice, solicitând adresa IP a sitului *bob.com*. Cum nu are nici o intrare pentru acest nume, serverul local interoghează serverul de nivel superior pentru domeniul *com* pentru a obține una. Totuși, Trudy o ia înaintea serverului *com* și trimite înapoi un răspuns fals spunând: „*bob.com* este 42.9.9.9" , unde acea adresă IP este a ei. Dacă răspunsul ei fals ajunge primul la ISP-ul lui Alice, va fi memorat în memoria ascunsă și răspunsul adevărat va fi respins ca răspuns nesolicitat la o cerere care nu mai e valabilă. A face un server DNS să instaleze o adresă IP falsă se numește **păcălirea DNS-ului** (eng.: **DNS spoofing**). O memorie ascunsă care conține o adresă IP intenționat falsă, ca aceasta, se numește **memorie ascunsă otăvită** (eng.: **poisoned cache**).

De fapt, lucrurile nu sunt chiar așa de simple. În primul rând, ISP-ul lui Alice verifică dacă răspunsul are adresa IP sursă corectă a serverului de nivel superior. Dar cum Trudy poate să pună orice dorește în acel câmp al mesajului, poate trece ușor de acest test, pentru că adresele IP ale serverelor de nivel superior trebuie să fie publice.

În al doilea rând, pentru ca serverele DNS să poată spune care răspuns corespunde cărei cereri, toate cererile poartă un număr de secvență. Pentru a păcăli ISP-ul lui Alice, Trudy trebuie să îi cunoască numărul de secvență curent. Cea mai ușoară metodă de a afla numărul de secvență este ca Trudy să–și înregistreze ea însăși un domeniu, să spunem *trudy-the-intruder.com*. Să presupunem că adresa IP a acestuia este tot 42.9.9.9. Ea creează și un server DNS pentru noul ei domeniu, să spunem *dns.trudy-the-intruder.com*. Și acesta folosește adresa IP a lui Trudy, 42.9.9.9, din moment ce Trudy are un singur calculator. Acum ea trebuie să-l facă pe ISP-ul lui Alice să înregistreze serverul ei DNS. Asta e ușor de făcut. Tot ceea ce are de făcut este să întrebe ISP-ul lui Alice de *foobar.trudy-the-intruder.com*, și ISP-ul lui Alice va afla cine se ocupă de noul domeniu al lui Trudy întrebând serverul *com* de nivel superior.

Cu *dns.trudy-the-intruder.com* aflat în siguranță în memoria ascunsă a ISP-ului lui Alice, adevăratul atac poate începe. Acum Trudy întreabă ISP-ul lui Alice de *www.trudy-the-intruder.com.* Bineînțeles, ISP-ul trimite serverului DNS al lui Trudy o cerere referitoare la acest domeniu. Această cerere poartă numărul de secvență pe care îl caută Trudy. Sprintenă ca un iepuraș, Trudy îi cere ISP-ului lui Alice să îl caute pe Bob. Apoi răspunde imediat la propria întrebare trimițând ISP-ului un răspuns falsificat, ca din partea serverului *com* de nivel superior, spunând: „*bob.com* este 42.9.9.9 ". Acest răspuns falsificat poartă un număr de secvență cu 1 mai mare decât cel pe care tocmai l-a primit ea. Dacă tot a ajuns aici, poate să mai trimită și un al doilea fals cu un număr de secvență cu 2 mai mare și poate încă o duzină de răspunsuri false cu numere de secvență consecutive. Unul dintre ele va trebui să se potrivească. Restul pur și simplu vor fi respinse. Când răspunsul fals ajunge la Alice, este stocat în memoria ascunsă; mai târziu, când va ajunge răspunsul adevărat, va fi respins pentru că nu va mai exista pentru el o cerere valabilă.

Acum, când Alice caută *bob.com*, i se spune să folosească 42.9.9.9, adresa lui Trudy. Trudy a organizat cu succes un atac omul-din-mijloc din confortabila ei cameră de zi. Diversii pași ai acestui atac sunt ilustrați în fig. 8-47. Pentru ca situația să fie și mai complicată, acesta nu e singurul mod de a păcăli DNS-ul. Mai există multe alte moduri.



1. Caută foobar.trudy-the-intruder.com (pentru a-i forța intrarea în memoria ascunsă a ISP-ului)
2. Caută www.trudy-the-intruder.com (pentru a afla următorul număr de secvență al ISP-ului)
3. Cere www. Trudy-the-intruder.com (Purtând următorul număr de secvență al ISP-ului, n)
4. Sprintenă ca un iepuraş, caută bob.com (pentru a forța ISP-ul să interogheze serverul com în pasul 5)
5. Legitimează cererea pentru bob.com cu numărul de secvență = n+1
6. Răspunsul fals al lui Trudy: Bob este 42.9.9.9, numărul de secvență = n+1
7. Răspunsul real (prea târziu)

Fig. 8-47. Cum păcălește Trudy ISP-ul lui Alice

DNS sigur

Acest atac, în particular, poate fi împiedicat făcând serverele DNS să folosească identificatori aleatorii în cererile lor în loc să numere pur și simplu, dar se pare că de fiecare dată când o gaură e astupată, se ivește una nouă. Adevărata problemă este că DNS-ul a fost proiectat într-o vreme în care Internetul era o facilitate pentru cercetare pentru câteva sute de universități și nici Alice, nici Bob, nici Trudy nu intraseră încă în joc. Pe atunci securitatea nu era o problemă; problema era de a face Internetul să funcționeze. Situația s-a schimbat radical cu trecerea anilor, așa că în 1994 IETF a înființat un grup de lucru care să facă DNS-ul fundamental sigur. Acest proiect este cunoscut sub numele de **DNSsec** (**DNS security,** rom.: securitatea DNS-ului); Rezultatele lui sunt prezentate în RFC 2535. Din nefericire, DNSsec nu a fost încă distribuit peste tot, așa că multe servere DNS sunt încă vulnerabile la atacuri de păcălire.

Conceptual, DNSsec este extrem de simplu. Este bazat pe criptografia cu chei publice. Fiecare zonă DNS (în sensul din figura7-4) are o pereche de chei formată dintr-una publică și una privată. Toate informațiile trimise de un server DNS sunt semnate cu cheia privată a zonei de origine, deci destinatarul poate să le verifice autenticitatea.

DNSsec oferă trei servicii fundamentale:

1. Dovada originii datelor.
2. Distribuția cheilor publice.
3. Autentificarea tranzacțiilor și a cererilor.

Principalul serviciu este primul dintre ele, care verifică faptul că datele returnate au fost aprobate de către deținătorul zonei. Al doilea este util pentru stocarea și refacerea, în siguranță, a cheilor publice. Al treilea este necesar pentru a proteja împotriva atacurilor prin reluare și a celor de păcălire. Observați că păstrarea secretului nu face parte dintre serviciile oferite deoarece toate informațiile din DNS sunt considerate publice. Cum etapele dezvoltării DNSsec se estimează că vor dura mai mulți ani, capacitatea serverelor ce pot asigura securitatea de a colabora cu cele ce nu o asigură este esențială, ceea ce implică faptul că protocolul nu poate fi schimbat. Să vedem acum niște detalii.

Înregistrările DNS sunt grupate în mulțimi numite **RRSets** (**Resource Record Sets**, rom.: Mulțimi de Înregistrări de Resurse), toate înregistrările cu același nume, clasă și tip fiind incluse într-o mulțime. O mulțime poate conține mai multe înregistrări A, de exemplu, dacă un nume DNS corespunde

unei adrese IP primare și unei adrese IP secundare. Mulțimile sunt extinse cu câteva tipuri noi de înregistrări (discutate mai jos). Pentru fiecare mulțime se face un cod de dispersie criptografic (de exemplu utilizând MD5 sau SHA-1). Rezumatul e semnat de către zona privată a RRSet. La primirea unei mulțimi semnate, clientul poate verifica dacă a fost semnată cu cheia privată a zonei de origine. Dacă semnătura coincide, datele sunt acceptate. Cum fiecare mulțime are propria semnătură, RRSet poate fi memorat oriunde, chiar pe un server neautorizat, fără să pună în pericol securitatea.

DNSsec introduce câteva tipuri noi de înregistrări. Primul din ele este înregistrarea *KEY*. Această înregistrare conține cheia publică a unei zone, utilizatorul, gazda, sau alt principal, algoritmul criptografic folosit pentru semnături, protocolul folosit pentru transmisie și încă biți în plus. Cheia publică este stocată ca atare. Certificatele X.509 nu sunt utilizate, din cauza dimensiunii lor. Câmpul corespunzător algoritmului este 1 pentru semnături MD5/RSA (varianta preferată), sau are alte valori pentru alte combinații. Câmpul corespunzător protocolului poate indica folosirea IPsec sau a altor protocoale de securitate, dacă s-a utilizat vreunul.

Al doilea tip nou de înregistrare este *SIG*. Acesta conține codul de dispersie semnat conform algoritmului specificat în înregistrarea *KEY*. Semnătura se aplică tuturor resurselor din set, inclusiv celor de tip *KEY*, dar nu și lui însuși. Înregistrarea mai conține și momentele când începe perioada de valabilitate a semnăturii și când aceasta expiră, precum și numele celui care semnează și alte câteva informații.

DNSsec este proiectat astfel încât cheia privată a unei zone să poată fi păstrată pe un calculator neconectat la rețea. O dată sau de două ori pe zi, conținutul bazei de date a unei zone poate fi transportat manual (de exemplu, pe CD-ROM) la o mașină neconectată unde se găsește cheia privată. Toate mulțimile de înregistrări pot fi semnate acolo și înregistrările SIG create astfel pot fi duse înapoi, pe CD-ROM, la serverul primar al zonei. Astfel, cheia privată poate fi memorată pe un CD-ROM încuiat într-un seif, mai puțin în momentele în care este introdus în calculatorul neconectat pentru a semna înregistrările noi. Când semnarea s-a terminat, toate copiile cheii sunt șterse din memorie și CD-ROM-ul este pus înapoi în seif. Această procedură reduce securitatea electronică la securitatea fizică, cu care oamenii știu cum să se descurce.

Această metodă de a pre-semna mulțimile de înregistrări accelerează foarte mult procesul de răspuns la cereri, deoarece nu mai trebuie făcute pe loc operații de criptografie. Dezavantajul este că e necesară o cantitate mare de spațiu pe disc pentru a stoca toate cheile și semnăturile în bazele de date DNS. Unele înregistrări își vor mări dimensiunea de zece ori din cauza semnăturii.

Când un proces client obține o mulțime de înregistrări semnată, trebuie să aplice cheia publică a zonei de origine pentru a decripta rezumatul, să calculeze el însuși rezumatul, și să compare cele două valori. Dacă sunt identice, datele sunt considerate valide. Totuși, această procedură ridică întrebarea cum va afla clientul cheia publică a zonei. Un mod este de a o cere unui server autorizat, folosind o conexiune sigură (de ex., folosind IPsec).

Totuși, în practică, e de așteptat ca clienții să fie reconfigurați cu cheile publice ale tuturor domeniilor de nivel superior. Dacă acum Alice vrea să viziteze situl Web al lui Bob, poate cere DNS-ului mulțimea de înregistrări pentru *bob.com*, care va conține adresa IP a acestuia și o înregistrare de tip *KEY* cu cheia publică a lui Bob. Această mulțime de înregistrări va fi semnată de către nivelul superior *com*, deci Alice îi poate verifica ușor validitatea. Un exemplu de ce ar putea conține această mulțime de înregistrări se află în fig. 8-48.

Numele domeniului	Timp de viață	Clasă	Tip	Valoare
bob.com	86400	IN	A	36.1.2.3
bob.com	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com	86400	IN	SIG	86947503A8B848F5272E53930C...

Fig. 8-48. Un exemplu de mulțime de înregistrări (RRSet) pentru *bob.com*. Înregistrarea *KEY* este cheia publică a lui Bob. Înregistrarea *SIG* este rezumatul serverului *com* de nivel superior pentru înregistrările A și *KEY* , pentru a le verifica autenticitatea.

Înarmată acum cu o copie verificată a cheii publice a lui Bob, Alice poate întreba serverul DNS al lui Bob (rulat de Bob) de adresa IP pentru *www.bob.com*. Această mulțime de înregistrări va fi semnată cu cheia privată a lui Bob, deci Alice poate verifica semnătura mulțimii de înregistrări pe care o returnează Bob. Dacă Trudy reușește în vreun fel să introducă o mulțime de înregistrări falsă în una din memoriile ascunse, Alice poate detecta cu ușurință lipsa ei de autenticitate pentru că înregistrarea *SIG* va fi incorectă.

Totuși, DNSsec oferă și un mecanism criptografic de a lega un răspuns de o anumită cerere, pentru a preveni înșelătoria pe care Trudy a reușit să o facă în fig. 8-47. Această măsură (opțională) de prevenire a înșelătoriilor adaugă la răspuns un cod de dispersie al mesajului de interogare semnat cu cheia privată a celui care răspunde. Cum Trudy nu știe cheia privată a serverului *com* de nivel superior, nu poate falsifica un răspuns la o cerere pe care ISP-ul lui Alice a trimis-o acolo. Desigur, ea poate să trimită prima răspunsul, dar acesta va fi respins din cauza semnăturii invalide a rezumatului cererii.

DNSsec suportă și alte câteva tipuri de înregistrări. De exemplu, înregistrarea *CERT* poate fi utilizată pentru a stoca certificate (de ex., certificate X.509). Această înregistrare a fost oferită pentru că unele persoane vor să transforme DNS-ul într-un PKI. Dacă aceasta se va întâmpla cu adevărat rămâne de văzut. Vom pune capăt aici discuției despre DNSsec. Pentru mai multe detalii, vă rugăm să consultați RFC 2535.

Nume cu auto-certificare

DNS sigur nu este singura posibilitate de a proteja numele. O abordare complet diferită este întâlnită în **Sistemul Sigur de Fișiere (Secure File System)** (Mazières et. al, 1999). În acest proiect, autorii au conceput un sistem de fișiere sigur, scalabil, de întindere foarte largă, fără a modifica DNS-ul (standard) și fără a folosi certificate sau a presupune existența unei PKI. În această secțiune vom arăta cum au putut fi aplicate ideile lor în cazul Web-ului. Prin urmare, în descrierea de mai jos vom folosi terminologia Web în loc de cea a sistemelor de fișiere, utilizată în lucrarea respectivă. Dar, pentru a evita orice confuzie, deși această schemă *ar putea* să fie aplicată Web-ului pentru a obține o securitate înaltă, ea nu este folosită în prezent și ar avea nevoie de modificări substanțiale ale software-ului pentru a fi introdusă.

Vom începe prin a presupune că fiecare server Web are o pereche de chei formată dintr-o cheie publică și una privată. Esența ideii este că fiecare URL conține un cod de dispersie criptografic pentru numele serverului și cheia publică a acestuia, ca parte a URL-ului. De exemplu, în fig. 8-49 vedem URL-ul pentru fotografia lui Bob. Începe cu obișnuita schemă *http*, urmată de numele DNS al serverului (*www.bob.com*). Apoi urmează două puncte și un rezumat de 32 de caractere. La sfârșit

este numele fișierului, din nou în modul obișnuit. Cu excepția codului de dispersie, acesta este un URL standard. Împreună cu codul, este un **URL cu auto-certificare** (eng.: **self-certifying URL**).

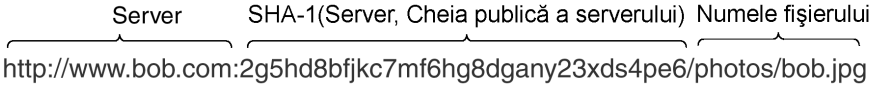


Fig. 8-49. Un URL cu auto-certificare conținând un rezumat al numelui serverului și al cheii publice.

Întrebarea evidentă este: La ce folosește codul de dispersie? Codul e calculat concatenând numele DNS al serverului cu cheia publică a acestuia și aplicând rezultatului funcția SHA-1 pentru a obține un cod pe 160 de biți. În această schemă, rezumatul e reprezentat ca o secvență de 32 de cifre și litere mici, cu excepția literelor „l" și „o" și a cifrelor „1" și „0", pentru a evita confuziile. Deci au mai rămas 32 de cifre și de litere posibile. Fiecare dintre cele 32 de caractere disponibile poate codifica un șir de 5 biți. Un șir de 32 de caractere poate reprezenta rezumatul SHA-1 de 160 de biți. De fapt, nu e necesară folosirea unui rezumat; ar putea fi folosită cheia însăși. Avantajul rezumatului este că reduce lungimea numelui.

Cel mai simplu (dar cel mai puțin convenabil) mod de a vedea fotografia lui Bob este ca Alice să trimită pur și simplu șirul din fig. 8-49 programului de navigare. Programul de navigare trimite un mesaj la situl lui Bob, cerându-i cheia publică. Când cheia publică a lui Bob ajunge, programul de navigare concatenează numele serverului și cheia publică și execută algoritmul de calcul al rezumatului. Dacă rezultatul coincide cu rezumatul de 32 de caractere din URL-ul securizat, programul de navigare este sigur că are cheia publică a lui Bob. Până la urmă, datorită proprietăților algoritmului SHA-1, chiar dacă Trudy interceptează cererea și falsifică răspunsul, nu are cum să găsească o cheie publică prin care să se obțină rezumatul așteptat. Orice interferență din partea ei va fi astfel detectată. Cheia publică a lui Bob poate fi stocată în memoria ascunsă pentru utilizări viitoare.

Acum Alice trebuie să verifice că Bob are cheia privată corespunzătoare. Ea construiește un mesaj conținând o cheie de sesiune AES propusă, un număr ad-hoc și o amprentă de timp. Pe urmă criptează mesajul cu cheia publică a lui Bob și i-l trimite. Cum doar Bob are cheia privată corespunzătoare, numai el poate să decripteze mesajul și să trimită înapoi numărul ad-hoc criptat cu cheia AES. Când primește numărul corect criptat AES, Alice știe că vorbește cu Bob. De asemenea, Alice și Bob au acum o cheie de sesiune AES pentru următoarele cereri și răspunsuri *GET*.

Odată ce Alice are fotografia lui Bob (sau orice altă pagină Web), ea poate să o marcheze (eng.: bookmark), ca să nu mai trebuiască să scrie încă o dată întregul URL. Mai mult, URL-urile incluse în paginile Web pot fi de asemenea cu auto-certificare, ca să poată fi folosite doar făcându-se un clic pe ele, dar având în plus și siguranța că pagina returnată este cea corectă. Alte moduri de a evita scrierea inițială a URL-urilor cu auto-certificare sunt obținerea lor printr-o conexiune sigură de la un server autorizat sau prezența lor în certificate X.509 semnate de autorități de certificare.

Altă metodă de a obține URL-uri cu auto-certificare ar fi conectarea la un motor de căutare autorizat, introducând (prima dată) URL-ul cu auto-certificare al acestuia, și trecerea prin același protocol cu cel descris mai sus, ceea ce va duce la o conexiune sigură, autentificată, cu motorul de căutare. Apoi motorul de căutare poate fi interogat, iar rezultatele vor apărea pe o pagină semnată, plină cu URL-uri auto-certificate ce pot fi accesate printr-un clic, fără a mai fi necesară scrierea unor șiruri lungi.

Acum să vedem cât de bine rezistă această abordare la păcălelile lui Trudy. Dacă Trudy reușește otrăvirea memoriei ascunse de la ISP-ul lui Alice, cererea lui Alice ar putea fi în mod greșit livrată lui Trudy în loc de Bob. Dar acum protocolul cere ca cel ce primește un mesaj inițial (adică Trudy) să returneze o cheie care să producă rezumatul corect. Dacă Trudy își trimite propria cheie publică, Alice o va detecta imediat pentru că rezumatul SHA-1 nu se va potrivi cu URL-ul auto-certificat. Dacă Trudy trimite cheia publică a lui Bob, Alice nu va detecta atacul, dar va cripta mesajul următor folosind cheia lui Bob. Trudy va primi mesajul, dar nu va putea să-l decripteze pentru a extrage cheia AES și numărul ad-hoc. În ambele cazuri, tot ceea ce poate face această păcălire a DNS-ului este să provoace un atac de refuz al serviciului.

8.9.3 SSL – Nivelul soclurilor sigure (Secure Sockets Layer)

Securitatea numelor pe Web reprezintă un bun start, dar există mult mai multe de spus despre securitatea pe Web. Următorul pas îl reprezintă conexiunile sigure. Acum vom discuta despre cum se poate ajunge la conexiuni sigure.

Atunci când Web-ul a intrat în atenția publicului, el era folosit pentru distribuția de pagini statice. Oricum, nu după mult timp, câteva companii au avut ideea de a-l folosi pentru tranzacții financiare, cum ar fi de exemplu cumpărarea de bunuri cu ajutorul cărții de credit, operațiuni bancare online și schimburi electronice de acțiuni. Aceste aplicații au creat o cerere pentru conexiuni sigure. În 1995, compania Netscape Communications, furnizorul dominant de navigatoare de web din acel moment, a răspuns acestei cereri prin introducerea unui pachet de securitate denumit **SSL** (***Secure Sockets Layer*** rom.: ***nivelul soclurilor sigure***). Acest program împreună cu protocolul său sunt larg folosite acum și de Internet Explorer, așa că merită să fie examinat mai în detaliu.

SSL-ul realizează o conexiune sigură între două socluri, precum și

1. Negocierea parametrilor între client și server
2. Autentificare mutuală a clientului și serverului
3. Comunicare secretă
4. Protecția integrității datelor

Am mai văzut aceste elemente și înainte astfel că nu este nevoie să insistăm asupra lor în continuare.

Poziția SSL-ului în cadrul stivei de protocoale este ilustrată în fig. 8-50. De fapt, este un nou nivel interpus între nivelul aplicație și cel de transport, acceptând cereri din partea unui program de navigare și trimițându-le mai jos către TCP pentru transmisia către server. Din momentul în care s-a realizat conexiunea sigură, principala sarcină a SSL-ului constă în asigurarea compresiei și a criptării. Când HTTP este folosit deasupra SSL-ului, este denumit **HTTPS** (rom.: *HTTP securizat – eng.: Secure HTTP*), deși este vorba de protocolul standard HTTP. Câteodată este disponibil la un nou port (443) în loc de portul standard (80). Ca o particularitate, folosirea SSL nu este restricționată numai cu programe de navigare Web, deși aceasta este aplicația cea mai des întâlnită.

Aplicație (HTTP)
Securitate (SSL)
Transport (TCP)
Rețea (IP)
Legătură de date (PPP)
Fizic (modem, ADSL, cablu TV)

Fig. 8-50. Niveluri (și protocoale) pentru un utilizator obișnuit ce navighează cu SSL.

Protocolul SSL a cunoscut mai multe versiuni. În cele ce urmează, vom discuta numai despre versiunea 3, care este cea mai folosită. SSL suportă o diversitate de algoritmi și opțiuni diferite. Aceste opțiuni includ prezența sau absența compresiei, algoritmii de criptare ce vor fi folosiți și câteva chestiuni legate de restricțiile de export impuse criptografiei. Cea din urmă este gândită în principal pentru a asigura că criptografia serioasă este folosită numai în cazul în care ambele capete ale conexiunii sunt situate în Statele Unite. În alte cazuri, cheile sunt limitate la 40 de biți, ceea ce pentru criptografi reprezintă o glumă. Netscape a fost forțat să impună această restricție pentru a putea obține o licență de export din partea Guvernului Statelor Unite.

SSL constă din două subprotocoale, unul pentru stabilirea unei conexiuni sigure și unul pentru folosirea acesteia. Să începem mai întâi să vedem cum sunt stabilite conexiunile securizate. Subprotocolul de stabilire a conexiunii este arătat în fig. 8-51. El începe cu mesajul 1 când Alice trimite o cerere către Bob pentru stabilirea unei legături. Cererea specifică versiunea de SSL pe care o are Alice și preferințele sale cu privire la algoritmii de compresie și de criptare. De asemenea, conține un număr ad-hoc (eng.: nonce), R_A, ce va folosit mai târziu.

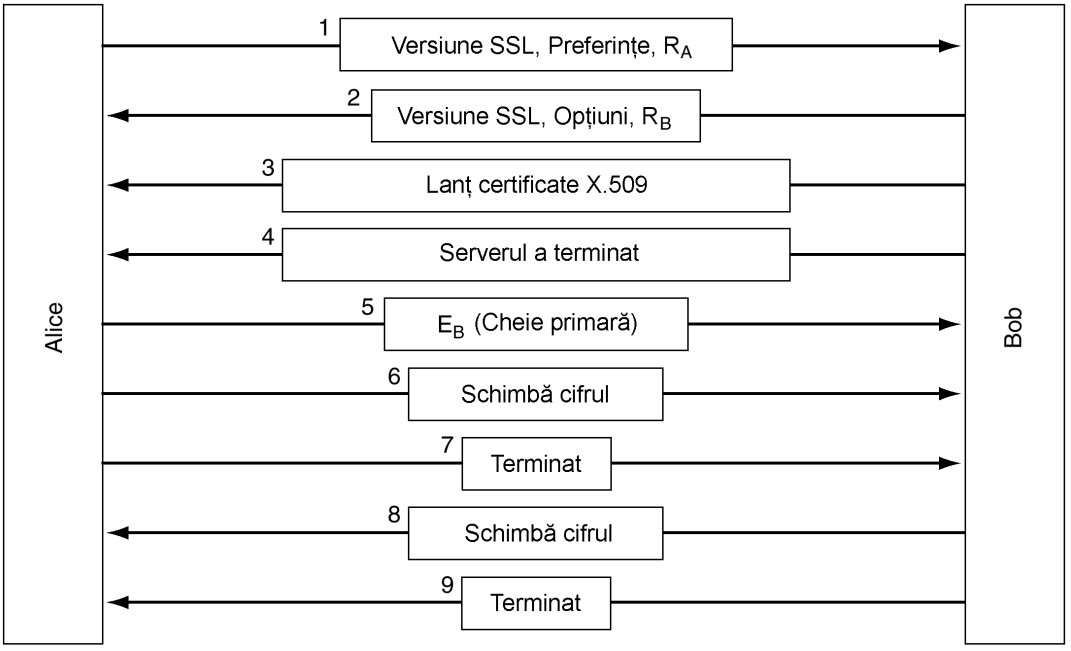


Fig. 8-51. O versiune simplificată a subprotocolului de stabilire unei conexiuni SSL

Acum este rândul lui Bob. În cel de al doilea mesaj, Bob face o alegere între diversii algoritmi pe care Alice îi poate suporta și își trimite numărul lui ad-hoc, R_B. Apoi, în mesajul 3, el trimite un certificat ce conține cheia lui publică. Dacă acest certificat nu este semnat de o autoritate recunoscută, el trimite de asemenea și un lanț de certificate ce pot fi urmate în sens invers până la ultimul. Toate programele de navigare, inclusiv cel al lui Alice, au predefinite aproximativ 100 de chei publice, așa că dacă Bob poate stabili un lanț care ajunge la una din acestea, Alice va putea să verifice cheia publică a lui Bob. În acest moment, Bob poate trimite și alte mesaje (cum ar fi o cerere pentru certificatul cheii publice al lui Alice). Când Bob a terminat, el trimite mesajul numărul 4 pentru a-i spune lui Alice că este rândul ei.

Alice răspunde prin alegerea unei **chei primare** (eng.: *premaster key*) aleatoare de 384 de biți și prin trimiterea acesteia lui Bob, criptată cu cheia lui publică (mesajul numărul 5). Cheia sesiunii folosită de fapt pentru criptarea datelor este derivată din cheia primară combinată cu ambele numere ad-hoc într-o manieră complexă. După ce mesajul 5 a ajuns, atât Alice cât și Bob sunt în stare să calculeze cheia sesiunii. Datorită acestui fapt, Alice îi spune lui Bob să folosească noul cifru (mesajul 6) și de asemenea că a terminat subprotocolul de stabilire (mesajul 7). Bob îi răspunde afirmativ la aceste mesaje (mesajele cu numerele 8 și 9).

Totuși, deși Alice știe cine este Bob, Bob nu știe cine este Alice (în afara cazului în care Alice are o cheie publică și un certificat corespunzător pentru ea, o situație puțin probabilă în cazul unui individ). De aceea, primul mesaj al lui Bob poate foarte bine să fie o cerere pentru ca Alice să se autentifice folosind un nume de login și o parolă ce au fost stabilite anterior. Protocolul de autentificare este în afara domeniului de acoperire al SSL-ului. De îndată ce a fost îndeplinit, prin orice mijloace, transportul de date poate începe.

Așa cum este menționat și mai sus, SSL suportă mai mulți algoritmi de criptare. Cel mai puternic dintre aceștia folosește triplu DES cu trei chei separate pentru criptare și SHA-1 pentru integritatea mesajului. Această combinație este relativ lentă, așa că este folosită de cele mai multe ori de operațiile bancare și alte aplicații în care este necesar cel mai mare nivel de securitate. Pentru aplicații obișnuite de comerț electronic este folosit RC4 cu o cheie de 128 de biți pentru criptare și MD5 pentru autentificarea mesajului. RC4 folosește cheia de 128 de biți ca un punct de plecare și o extinde la un număr mult mai mare pentru uzul intern. Apoi folosește acest număr intern pentru a genera un șir-cheie. Acesta este supus unei operații SAU EXCLUSIV cu textul pentru a genera un flux de cifru clasic, așa cum am văzut în fig. 8-14. Versiunile de export folosesc de asemenea RC4 cu chei de 128 de biți, dar 88 de biți dintre aceștia sunt făcuți public pentru a face cifrul mai ușor de spart.

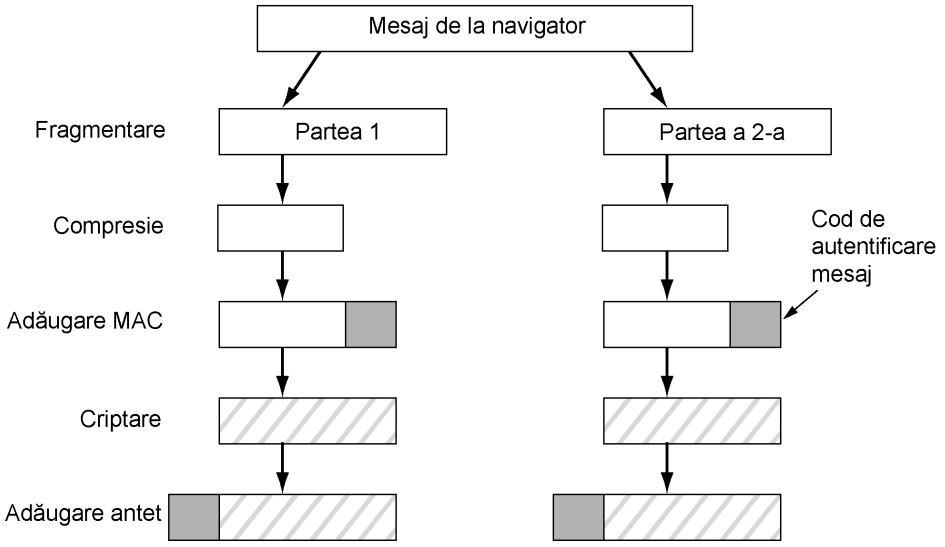


Fig. 8-52. Transmisie de date folosind SSL

Pentru transportul efectiv, este folosit un al doilea subprotocol, așa cum este arătat în fig. 8-52. Mesajele de la navigator sunt mai întâi sparte în bucăți de până la 16 KB. Dacă compresia este activată, atunci fiecare unitate este apoi compresată. După aceea, o cheie secretă derivată din cele două

numere ad-hoc și cheia primară este concatenată cu textul compresat și rezultatul este rezumat cu algoritmul de dispersie agreat (de obicei MD5). Acest cod de dispersie este adăugat fiecărui fragment ca MAC (*Message authentication code – cod pentru autentificarea mesajului*). Fragmentul compresat împreună cu MAC este apoi criptat cu algoritmul de criptare simetrică agreat (de obicei prin aplicarea operației SAU EXCLUSIV cu șirul-cheie RC4). În final, este atașat un antet de fragment și fragmentul este transmis prin intermediul conexiunii TCP.

Prudența este totuși importantă. Deoarece a fost demonstrat că RC4 are câteva chei slabe ce pot fi analizate criptologic, securitatea SSL folosind RC4 este pe un teren șubred (Fluhrer et. al, 2001). Programele de navigare ce permit utilizatorului să aleagă suita de cifruri ar trebui configurate să folosească mereu DES triplu cu chei de 168 de biți și SHA-1, chiar dacă această combinație este mai lentă decât RC4 și MD5.

O altă problemă cu SSL o reprezintă faptul că principalii pot să nu aibă certificate și chiar dacă au, ei nu verifică întotdeauna dacă cheile folosite se potrivesc cu certificatele.

În 1996, compania Netscape Communications a trimis SSL către IETF pentru standardizare. Rezultatul a fost **TLS** (eng.: *Transport Layer Security – rom.: Securitate la nivelul de transport*). Acesta este descris în RFC 2246.

Schimbările aduse SSL au fost relativ mici, dar suficiente pentru ca versiunea 3 de SSL și TLS să nu poată coopera. De exemplu, modul în care cheia de sesiune este derivată din cheia primară (eng.: premaster key) și numerele ad-hoc a fost schimbat pentru a face cheia mult mai puternică (adică mai greu de criptanalizat). Versiunea de TLS este cunoscută ca versiunea 3.1 SSL. Prima implementare a apărut în anul 1999, dar încă nu este clar dacă TLS va înlocui SSL în practică, deși este puțin mai puternic. Problema cu cheile slabe RC4 rămâne totuși.

8.9.4 Securitatea codului mobil

Denumirile și conexiunile sunt două zone de interes pentru securitatea pe Web. Dar sunt mai multe. La început, când paginile Web erau doar fișiere statice HTML, ele nu conțineau cod executabil. Acum, ele conțin adesea mici programe, inclusiv applet-uri Java, programe ActiveX și fragmente JavaScript. Descărcarea și execuția de **cod mobil** (eng.: *mobile code*) prezintă un risc de securitate mare; de aceea au fost elaborate diverse metode pentru micșorarea acestui risc. Vom analiza acum pe scurt unele probleme ce privesc codul mobil și câteva dintre soluțiile de rezolvare.

Securitatea applet-urilor Java

Applet-urile Java sunt mici programe Java care se traduc în limbajul unei mașini cu stivă denumit **JVM** (eng.: *Java Virtual Machine* – rom.: *Mașină virtuală Java*). Ele pot fi puse intr-o pagină Web fiind descărcate împreună cu pagina. După ce pagina a fost descărcată, applet-urile sunt date unui interpretor JVM din programul de navigare, așa cum este ilustrat în fig. 8-53.

Avantajul execuției codului interpretat față de codul compilat este că fiecare instrucțiune este examinată de către interpretor înaintea execuției. Astfel este dată o șansă interpretorului de a verifica dacă adresa instrucțiunii este validă. În plus, apelurile de sistem sunt, de asemenea, captate și interpretate. Felul în care aceste apeluri sunt manipulate este conform politicii locale de securitate. De exemplu, dacă un applet este de încredere (de exemplu, provine de pe discul local), apelurile de sistem pot fi executate fără probleme. Totuși, dacă un applet este nesigur (de exemplu, a venit din Internet), el poate fi încapsulat în ceea ce se numește un **mediu protejat** (eng.: *sandbox*) pentru a restricționa comportamentul său și a controla încercările sale de a folosi resursele sistemului.

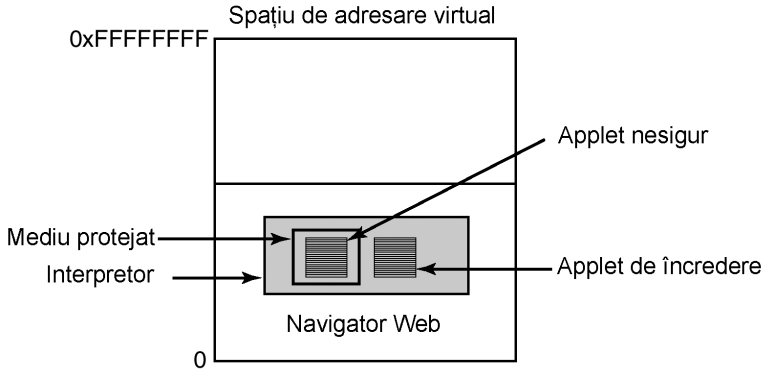


Fig. 8-53. Applet-urile pot fi interpretate de către un navigator Web

Când un applet încearcă să folosesc resursele sistemului, apelul său este trimis pentru aprobare către un monitor de securitate. Acest monitor examinează apelul din punctul de vedere al politici locale de securitate și apoi decide dacă îl va permite sau îl va respinge. Astfel, este posibil să dea unor applet-uri acces la resurse, dar nu tuturor. Din nefericire, realitatea este că modelul de securitate lucrează prost și că defectele de programare apar mai tot timpul.

ActiveX

Controalele ActiveX sunt programe binare Pentium ce pot fi incluse în pagini Web. Când este întâlnit unul dintre ele, se efectuează o verificare pentru a vedea dacă ar trebui să fie executat sau nu, și este executat dacă trece testul. El nu este interpretat și nici nu este executat într-un mediu protejat, astfel încât codul are aceeași putere ca orice alt program și poate face mult rău. Astfel, întreaga securitate constă în a decide dacă controlul ActiveX va fi sau nu rulat.

Metoda aleasă de Microsoft pentru stabilirea acestei deciziei este bazată pe ideea de **semnare a codului** (eng.: *code signing*). Fiecare program ActiveX vine împreună cu o semnătură digitală – un rezumat al codului ce este semnat de cel ce a creat codul folosind criptografia cu cheie publică. Când un program ActiveX își face apariția, programul de navigare verifică semnătura pentru a se asigura că nu a suferit modificări în timpul transferului. Dacă semnătura este corectă, programul de navigare verifică tabelele sale interne pentru a vedea dacă cel ce a creat programul este de încredere sau există un lanț ce se termină la un creator de încredere. Dacă creatorul este de încredere, programul este executat; altfel, nu este executat. Sistemul Microsoft care verifică programele ActiveX este numit **Autenticod** (eng.: *Authenticode*).

Este bine să facem o comparație între modurile de abordare pentru Java și ActiveX. Cu Java, nu este important cine a scris applet-ul. În schimb, un interpretor asigură că nu face lucruri interzise de proprietarul mașinii. Prin contrast, cu semnarea codului, nu există nici o încercare de monitorizare a comportamentului codului mobil la execuție. Dacă el vine de la o sursă de încredere și nu a fost modificat pe parcurs, el este executat. Nu este făcută nici o încercare de a vedea dacă codul este răuvoitor sau nu. Dacă programatorul original *a* dorit să formateze discul dur și să șteargă memoria ROM flash, astfel încât calculatorul să nu mai poată fi pornit, și dacă programatorul a fost recunoscut ca fiind de încredere, codul va rula și va distruge calculatorul (excepție făcând cazul în care controalele ActiveX au fost dezactivate în programul de navigare).

Mulți oameni simt că încrederea acordată companiilor de programare necunoscute provoacă teamă. Pentru a demonstra problema, un programator din Seattle a format o companie de progra-

mare și a reușit să o facă recunoscută ca demnă de încredere, ceea ce este foarte ușor de făcut. Apoi a scris un program ActiveX ce realiza oprirea calculatorului și l-a distribuit. A oprit multe mașini, dar care au fost ulterior re-pornite astfel încât nu au fost pagube. Programatorul a încercat doar să pună în evidență problema. Răspunsul oficial a constat în suspendarea certificatului pentru acest program ActiveX, ceea ce a încheiat un scurt episod destul de jenant, dar problema nu a dispărut și poate fi încă exploatată de către un programator diabolic (Garfinkel cu Spafford, 2002). Deoarece nu există nici o cale de a supraveghea mii de companii de programare ce ar putea să scrie cod mobil, tehnica semnării codului este un dezastru ce așteaptă să se producă.

JavaScript

JavaScript nu are nici un model formal de securitate, dar are o lungă istorie de implementări vulnerabile. Fiecare furnizor tratează problema securității într-o manieră diferită. De exemplu, versiunea 2 de Netscape Navigator a folosit ceva apropiat de modelul Java, dar la versiunea 4 a fost abandonată această manieră în favoarea modelului de cod semnat.

Problema fundamentală este că permițând execuția de cod străin pe mașina dumneavoastră vă expuneți multor necazuri. Din punct de vedere al securității, este ca și cum invitați un hoț în casa dumneavoastră și apoi încercați să-l urmăriți cu atenție pentru a nu putea ieși din bucătărie și intra în sufragerie. Dacă se întâmplă ceva neașteptat și sunteți neatent pentru o clipă, se pot întâmpla lucruri rele. Problema este că codul mobil permite o grafică atractivă și o interacțiune rapidă și foarte mulți proiectanți Web cred că acestea sunt mult mai importante decât securitatea, în special când este vorba de riscul altcuiva.

Virușii

Virușii sunt o altă formă de cod mobil. Spre deosebire de exemplele de mai sus, virușii nu sunt invitați deloc. Diferența între un virus și un cod mobil obișnuit este aceea că virușii sunt scriși pentru a se reproduce. Când un virus ajunge într-un calculator, prin intermediul unei pagini Web, al unui document atașat sau printr-o altă cale, el începe prin infectarea programelor executabile de pe disc. Când unul dintre aceste programe este executat, controlul este transferat virusului, care de obicei încearcă să se întindă pe alte mașini, de exemplu, prin trimiterea de mesaje cu propriile sale copii către toate persoanele din agenda victimei. Unii viruși infectează sectorul de start al discului dur, astfel că atunci când mașina pornește, virusul este activat. Virușii au devenit o problemă uriașă pe Internet și au provocat pagube de miliarde de dolari. Nu există o soluție foarte clară. Probabil că soluția ar fi cea a unei noi generații de sisteme de operare bazate pe micronuclee și o separare puternică a utilizatorilor, proceselor și resurselor.

8.10 IMPLICAȚII SOCIALE

Internet-ul și tehnologiile sale de securitate reprezintă o zonă în care problemele de ordin social, politici publice și tehnologie se întâlnesc într-o confruntare directă, deseori cu consecințe grave. În cele ce urmează vom analiza pe scurt trei zone: confidențialitate, libertatea de exprimare și dreptul de autor. Nu mai este nevoie să spunem că atingem doar tangențial aceste subiecte. Pentru lecturi suplimentare, a se vedea (Anderson, 2001; Garfinkel cu Spafford, 2002; Schneier, 2000). Internet-ul este, de asemenea, plin de materiale documentare. Este suficient să folosiți cuvinte cheie precum

„confidențialitate” (eng.: *privacy*), „cenzură” (eng.: *censorship*) și „drept de autor” (eng.: *copyright*) în orice motor de căutare. De asemenea puteți consulta pagina Web a acestei cărți pentru alte câteva legături interesante.

8.10.1 Confidențialitate

Au oamenii dreptul la confidențialitate? Bună întrebare. Cel de-al patrulea amendament al Constituției Statelor Unite împiedică guvernul de-a scotoci casele, documentele și obiectele personale ale oamenilor fără un motiv întemeiat și stabilește circumstanțele în care sunt emise mandatele de percheziție. Astfel, confidențialitatea a fost o problemă publică de peste 200 de ani, cel puțin în Statele Unite.

Ceea ce s-a schimbat în ultimul deceniu sunt atât ușurința cu care guvernele își pot spiona cetățenii, cât și ușurința cu care cetățenii pot preveni un astfel de spionaj. În secolul al XVIII-lea, pentru ca guvernul să cerceteze documentele unui cetățean trebuia să trimită un polițist călare la ferma cetățeanului pentru a vedea anumite documente. Era o procedură greoaie. În zilele noastre, companiile de telefonie și furnizorii de Internet instalează dispozitive de ascultare atunci când le sunt prezentate mandate de percheziție. Viața este mult mai ușoară astfel pentru polițist și nici numai există pericolul unei căzături de pe cal.

Criptografia schimbă toate aceste aspecte. Oricine își permite deranjul de a descărca și instala PGP și folosește o cheie bine păzită de o putere extraterestră poate fi destul de sigur că nimeni în universul pe care-l cunoaștem nu îi poate citi mesajele electronicele, cu sau fără mandat de percheziție. Guvernele înțeleg foarte bine această chestiune și nu o agreează. Confidențialitate reală înseamnă dificultăți mai mari în a spiona criminalii de orice fel, dar și greutăți în a spiona jurnaliști și oponenți politici. Prin urmare, unele guverne restricționează sau interzic folosirea sau exportul de criptografie. În Franța, de exemplu, până în 1999, toată criptografia era interzisă dacă cheile nu erau date de către guvern.

Franța nu a fost o excepție. În aprilie 1993, guvernul Statelor Unite a anunțat intenția sa de realiza un criptoprocesor hardware, **cip-ul clipper** (eng.: *clipper chip*), standard pentru toate comunicațiile în rețea. În acest fel, s-a spus, confidențialitatea cetățenilor va fi garantată. De asemenea era menționat că cip-ul permitea guvernului decriptarea traficului prin intermediul unei scheme numite **custodie de chei** (eng.: *key escrow*), ce permitea guvernului să aibă acces la toate cheile. Totuși, se promitea ascultarea mesajelor numai cu un mandat valabil de percheziție. Nu mai trebuie spus că a urmat o largă dispută, avocații confidențialității denunțând tot planul și avocații oficiali lăudându-l. În cele din urmă, guvernul a bătut în retragere și a renunțat la idee.

O mare cantitate de informație despre confidențialitatea electronică poate fi găsită pe pagina Web a Fundației Graniței Electronice (eng.: Electronic Frontier Foundation), *www.eff.org*.

Retransmițătoare anonime

PGP, SSL și alte tehnologii fac posibil ca două părți să stabilească o comunicație securizată, autentificată, neurmărită de o terță parte și fără interferențe. Totuși, câteodată, confidențialitatea este cel mai bine servită de absența autentificării, prin realizarea unei comunicații anonime. Anonimitatea poate fi dorită pentru mesaje punct-la-punct, grupuri de știri sau în ambele cazuri.

Să considerăm câteva exemple. În primul rând, dizidenții politici ce trăiesc sub regimuri autoritare doresc deseori să comunice în mod anonim pentru a nu fi trimiși la închisoare sau omorâți. În al doilea rând, comportamentul ilegal din foarte multe corporații, organizații educaționale, guverna-

mentale și altele a fost făcut public de către cei interesați, care de cele mai multe ori preferă să rămână anonimi pentru a evita plătirea eventualelor polițe ulterioare. În cel de al treilea rând, oamenii cu vederi sociale, politice sau religioase nepopulare ar dori să comunice unul cu altul prin intermediul mesajelor electronice sau a grupurilor de știri fără a se expune. În al patrulea rând, oamenii ar putea dori să discute despre alcoolism, boli mentale, hărțuire sexuală, abuzul asupra copiilor sau despre faptul de a fi membru a unei minorități persecutate politic fără a-și face publice părerile. Există, bineînțeles, numeroase alte exemple.

Să luăm în considerare un exemplu concret. În anii 1990, câțiva critici ai unui grup religios netradiționalist și-au publicat părerile pe un grup de știri USENET prin intermediul unui **retransmițător anonim** (eng.: *anonymous remailer*). Acel server permitea utilizatorilor să creeze pseudonime și să trimită mesaje server-ului, care apoi le retransmitea și le republica folosind pseudonimul, astfel încât nimeni nu și-a dat seama din partea cui au venit. Unele dintre mesaje arătau ceea ce grupul religios pretindea a fi secrete ale meseriei și documente cu drept de autor. Grupul religios a răspuns comunicând autorităților locale că secretele meseriei sale au fost făcute publice și că drepturile de autor au fost încălcate, acestea fiind delicte în zona în care server-ul era localizat. A urmat un proces și operatorul server-ului a fost obligat să predea informația care a dezvăluit identitățile persoanelor ce au publicat anunțurile (în mod întâmplător, aceasta nu era prima dată când religia nu era fericită când cineva i-a descoperit secretele: William Tyndale a fost ars pe rug în 1536 pentru că a tradus Biblia în limba engleză).

O destul de mare parte a comunității Internet a fost scandalizată de o asemenea încălcare a confidențialității. Concluzia pe care fiecare a tras-o a fost că un retransmițător anonim care menține o tabelă de corespondențe între adrese de poștă reale și pseudonime (numit retransmițător de tipul 1) nu valorează prea mult. Acest caz a stimulat diverse persoane să conceapă retransmițătoare anonime ce ar putea rezista atacurilor cu citație (eng.: *subpoena attacks*).

Aceste retransmițătoare noi, deseori denumite **retransmițătoare de tip cypherpunk** (eng.: *cypherpunk remailers*), funcționează după cum urmează. Utilizatorul produce un mesaj electronic, îl completează cu antetele specifice RFC 822 (exclusiv câmpul *De la:* - eng.: *From:*), îl criptează cu cheia publică a retransmițătorului și îl trimite retransmițătorului. Acolo, antetele externe RFC 822 sunt șterse, conținutul este decriptat și mesajul este retransmis. Retransmițătorul nu are conturi și nici nu păstrează fișiere cu istoria lucrurilor petrecute (eng.: *log-uri*), așa că, chiar dacă server-ul este mai târziu confiscat, el nu reține nici o urmă a mesajelor ce au trecut prin el.

Mulți utilizatori ce doresc să rămână anonimi își revendică cererile prin intermediul mai multor retransmițătoare, așa cum este ilustrat în fig. 8-54. Astfel, Alice dorește să-i trimită lui Bob o felicitare de ziua Sfântului Valentin, foarte, foarte, foarte anonimă, astfel că folosește trei retransmițătoare. Ea compune mesajul, M și îi pune un antet ce conține adresa lui Bob. Apoi îl criptează cu cheia publică a retransmițătorului 3, E_3 (indicat prin liniile orizontale). La acesta ea prefixează un antet cu adresa retransmițătorului 3 în text clar. Acest mesaj este arătat între retransmițătoarele 2 și 3.

Apoi ea criptează mesajul cu cheia publică a retransmițătorului 2, E_2 (indicat prin liniile verticale) și adăugă la început un antet de text clar ce conține adresa retransmițătorului 2. Acest mesaj este arătat în fig. 8-54 între 1 și 2. În cele din urmă, criptează întregul mesaj cu cheia publică a retransmițătorului 1, E_1, și pune la începutul lui un antet de text clar cu adresa retransmițătorului 1. acest mesaj este arătat în figură la dreapta lui Alice și este ceea ce ea transmite de fapt.

Când mesajul ajunge la retransmițătorul 1, antetul exterior este șters. Corpul mesajului este decriptat și apoi trimis retransmițătorului 2. Pași similari se petrec la nivelul celorlalte două retransmițătoare.

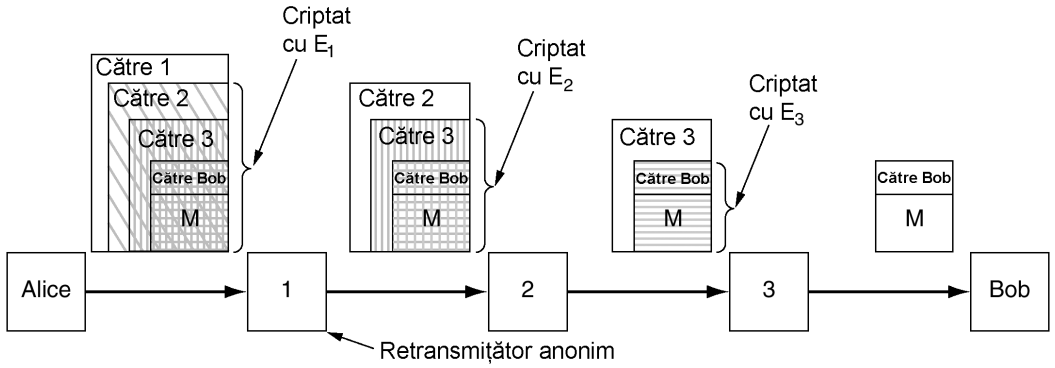


Fig. 8-54. Alice folosește 3 retransmițătoare pentru a trimite lui Bob un mesaj.

Deși este extrem de dificil pentru cineva să urmărească mesajul final pe calea de retur către Alice, multe retransmițătoare își iau măsuri suplimentare de siguranță. De exemplu, ele pot păstra mesajele pentru o perioadă aleatoare de timp, pot adăuga sau șterge bucăți nefolositoare la sfârșitul mesajului, pot reordona mesajele, toate acestea pentru a face dificil oricui să determine ce mesaj de ieșire al unui retransmițător corespunde unui mesaj de intrare, pentru a îngreuna analiza traficului. Pentru o descriere amănunțită a unui sistem ce reprezintă o capodoperă în mesageria anonimă, vedeți (Mazières și Kaashoek, 1998).

Anonimitatea nu este restricționată la mesageria electronică. Există servicii ce permit navigare anonimă pe Web. Utilizatorul își configurează programul de navigare pentru a folosi serviciul anonim ca pe un proxy. De aici înainte toate cererile HTTP se vor duce la serviciul anonim, care va cere pagina și o va trimite înapoi. Pagina Web va vedea că serviciul de „anonim”-izare este sursa a cererii, și nu utilizatorul. Atâta timp cât serviciul de anonimizare nu va ține un istoric al activității, nimeni nu își poate da seama cine a cerut pagina.

8.10.2 Libertatea de exprimare

Confidențialitatea se referă la indivizii ce doresc să limiteze accesul celorlalți la informații despre propria lor persoană. O a doua problemă socială cheie o reprezintă libertatea de exprimare și opusa ei, cenzura, care are legătură cu guvernele ce doresc să restricționeze ceea ce indivizii pot citi și publica. Deoarece conține milioane și milioane de pagini, Web-ul a devenit paradisul cenzurii. Materialul interzis, în funcție de natura și ideologia regimului, poate include pagini Web ce conțin elemente din lista următoare:

1. Materiale nepotrivite pentru copii și adolescenți.
2. Ură față de diverse grupări etnice, religioase, sexuale și altele.
3. Informații despre democrație și valori democratice.
4. Relatări ale unor evenimente istorice ce contrazic versiunea guvernamentală.
5. Manuale pentru deschiderea încuietorilor, construirea de arme, criptare de mesaje, etc.

Măsura obișnuită este de a interzice paginile rele.

Câteodată, rezultatele sunt neașteptate. De exemplu, câteva biblioteci publice au instalat filtre Web pe calculatoarele lor pentru a le face prietenoase față de copii, interzicând paginile pornografice. Filtrele înscriu paginile pe listele lor negre, dar în același timp caută cuvinte nepotrivite în pagini-

le ce urmează a fi afișate. Într-un caz din comitatul Loudoun din Virginia, filtrul a blocat căutarea unor materiale despre cancerul de sân deoarece filtrul a văzut cuvântul „sân". Patronul bibliotecii a dat în judecată comitatul Loudoun. Pe de altă parte, în Livermore din California, un părinte a dat în judecată biblioteca publică pentru că *nu* a instalat un filtru, după ce băiatul ei de 12 ani a fost prins uitându-se la materiale pornografice. Ce ar trebui să facă o bibliotecă?

A scăpat din vederea multor oameni că World Wide Web este o rețea mondială. Nu toate țările sunt de acord cu ceea ce ar trebui să fie permis pe Web. De exemplu, în noiembrie 2000, un tribunal francez a dictat corporației californiene Yahoo să blocheze accesul utilizatorilor francezi la licitațiile de obiecte asociate nazismului de pe pagina Yahoo, deoarece astfel de materiale violează legea franceză. Yahoo a apelat la un tribunal nord-american, care a luat partea corporației, dar problema cu privire la „ce legi se aplică și unde se aplică" este departe de a fi rezolvată.

Imaginați-vă ce s-ar putea întâmpla dacă un tribunal din Utah ar spune Franței să blocheze paginile Web despre vin, deoarece nu se conformează cu legile mult mai stricte privitoare la alcool ale statului Utah? Presupuneți că China ar cere ca toate paginile Web despre democrație să fie interzise deoarece nu sunt în interesul statului. Oare legile iraniene referitoare la religie se aplică mult mai liberalei Suedii? Poate Arabia Saudită bloca pagini Web ce vorbesc de drepturile femeilor? Toată problema este o veritabilă cutie a Pandorei.

Un comentariu relevant al lui John Gilmore este: „Rețeaua interpretează cenzura ca pe o pagubă și o ocolește". Pentru o implementare concretă, uitați-vă la **serviciul etern** (eng.: *eternity service*) (Anderson, 1996). Scopul lui este de a se asigura că o informație publicată nu poate fi retrasă sau rescrisă, așa cum era destul de obișnuit în Uniunea Sovietică în timpul conducerii staliniste. Ca să folosească serviciul eternității, utilizatorul trebuie să specifice cât timp trebuie păstrat materialul, să plătească o taxă proporțională cu această durată și cu volumul materialului și să-l încarce apoi în Internet. După aceea, nimeni nu-l poate șterge sau edita, nici chiar și cel care l-a încărcat.

Cum ar putea fi implementat un asemenea serviciu? Modelul cel mai simplu este folosirea unui sistem de la egal la egal (eng.: *peer-to-peer*) în care documentele să fie depuse în câteva zeci de servere, fiecare primind o parte a taxei și, prin asta, un motiv să se alăture sistemului. Serverele ar trebui împărțite în cât mai multe jurisdicții legale pentru a obține o flexibilitate maximă. Liste de câte 10 servere selectate aleator ar fi păstrate în siguranță în mai multe locuri, astfel încât dacă unele sunt compromise, altele ar continua să furnizeze informațiile cerute. O autoritate ce dorește distrugerea documentului nu poate fi niciodată sigură că a găsit toate copiile. De asemenea, sistemul s-ar putea regenera: dacă află că unele copii au fost distruse, siturile rămase ar încerca să găsească noi locuri.

Serviciul eternității a fost prima propunere de sistem rezistent la cenzură. De atunci, au fost propuse și, uneori implementate și alte sisteme. Au fost adăugate diverse caracteristici noi, cum ar fi criptarea, anonimitatea și toleranța la defecte. Deseori, fișierele ce trebuie stocate sunt sparte în mai multe părți, fiecare memorată pe mai multe servere. Câteva exemple sunt Freenet (Clarke et. al, 2002), PASIS (Wylie et. al, 2000) și Publius (Waldman et. al, 2000). Alte rezultate sunt prezentate în (Serjantov, 2002).

Tot mai multe țări încearcă să impună reguli pentru exportul de lucruri imateriale, care includ pagini Web, programe, lucrări științifice, mesaje electronice, suport telefonic și altele. Chiar și Marea Britanie, care are o îndelungată tradiție a libertății de exprimare, se gândește la legi mult mai restrictive, care ar defini, de exemplu, discuțiile tehnice dintre un profesor britanic și studentul său străin la Universitatea din Cambridge ca pe un export legiferat ce are nevoie de o licență guvernamentală (Anderson, 2002). Nu este nevoie să mai spunem că asemenea politici sunt controversate.

Steganografie

În țările cu o cenzură puternică, dizidenții încearcă deseori să folosească tehnologia pentru a eluda cenzura. Criptografia permite trimiterea (nu întotdeauna legală) de mesaje secrete, dar dacă guvernul crede că Alice este o Persoană Rea, simplul fapt că ea comunică cu Bob ar putea să-l pună și pe el în aceeași categorie, deoarece guvernele represive recunosc conceptul de tranzitivitate, chiar dacă nu au prea mulți matematicieni. Retransmițătoarele anonime pot ajuta, dar dacă sunt interzise pe plan intern și mesajele către retransmițătoare externe necesită licențe de export, nu pot ajuta chiar atât de mult. Dar Web-ul poate.

Oamenii care vor să comunice în secret încearcă deseori să ascundă că de fapt comunicare are loc. Știința ascunderii mesajelor este denumită **steganografie** (*steganography*), din cuvintele grecești ce s-ar traduce prin „scriere acoperită”. De fapt, grecii antici o foloseau și ei. Herodot a scris despre un general care a ras în cap un sol, i-a tatuat pe scalp un mesaj și l-a lăsat să-i crească părul la loc înainte de a-l trimite în misiune. Tehnicile moderne sunt, conceptual, aceleași, doar că au o mai mare lărgime de bandă și o întârziere mai mică.

Ca un caz în speță, să considerăm fig. 8-55 (a). Această fotografie, realizată în Kenya, conține trei zebre care contemplă un arbore acacia. În fig. 8-55 (b) se pare că ar fi aceleași trei zebre și arborele acacia, dar există o atracție specială. Conține textul complet, necenzurat a cinci piese shakespeariene: *Hamlet*, *Regele Lear*, *Macbeth*, *Negustorul din Veneția* și *Iulius Cezar*. Împreună, aceste piese totalizează peste 700 kiloocteți de text.

(a) (b)

Fig. 8-55. (a) Trei zebre și un arbore. (b) Trei zebre, un arbore și textul complet a cinci piese de William Shakespeare.

Cum funcționează acest canal steganografic? Imaginea color originală este de 1024 x 768 pixeli. Fiecare pixel constă din trei numere de 8 biți, câte unul pentru fiecare din culorile roșu, verde și albastru a acelui pixel. Culoarea acelui pixel este formată de o superpoziție a celor trei culori. Metoda de codificare steganografică folosește bitul cel mai puțin semnificativ din fiecare octet al a culorilor RGB ca un canal ascuns. Astfel fiecare pixel are spațiu pentru 3 biți de informație secretă, unul în valoare roșie, unul în verde și unul în cea albastră. Într-o imagine de asemenea mărime, se pot memora până la 1024 x 768 x 3 biți sau 294.912 octeți de informație secretă.

Textul complet al celor cinci piese împreună cu o scurtă notiță însumează 734.891 octeți. Acest text a fost mai întâi comprimat la aproximativ 270 KB folosind un algoritm de compresie standard.

Rezultatul a fost apoi criptat folosind IDEA și inserat în biții cei mai puțin semnificativi ai fiecărui octet de culoare. După cum se poate vedea (de fapt, nu se poate vedea), existența informației este total invizibilă. Este la fel de invizibilă în versiunea mărită, color a imaginii. Ochiul nu poate diferenția ușor culoarea pe 21 de biți față de cea pe 24 de biți.

Figurând cele două imagini în alb-negru la o rezoluție scăzută nu demonstrează cât de puternică este această. Pentru a simți mai bine cum lucrează steganografia, autorul a pregătit o demonstrație, cu imaginea color la o rezoluție mare a fig. 8-55(b) având cele cinci piese incluse în ea. Demonstrația, inclusiv instrumentele pentru inserarea și extragerea textelor în și din imagini, poate fi găsită la pagina Web a cărții.

Pentru a folosi steganografia pentru comunicația nedetectată, dizidenții ar putea crea o pagină Web plină de imagini politic-corecte, cum ar fi fotografii ale Marelui Lider, sporturi locale, vedete de film și televiziune, etc. Desigur, imaginile ar fi pline de mesaje steganografice. Dacă mesajele ar fi întâi compresate și apoi criptate, chiar și cineva care le-ar suspecta prezența ar avea dificultăți imense în distingerea mesajelor față de zgomot. Desigur, imaginile ar trebui să fie scanări recente; copiind o imagine de pe Internet și schimbând o parte din biți nu conduc la rezultat.

Imaginile nu sunt singurele purtătoare de mesaje steganografice. Fișierele video sunt de asemenea bune. Fișierele video au o lărgime de bandă steganografică foarte mare. Chiar și așezarea și ordinea etichetelor într-un fișier HTML pot fi purtătoare de informație.

Deși am examinat steganografia în contextul exprimării libere, ea are numeroase alte utilizări. O utilizare banală este codificarea drepturilor de proprietate chiar în imaginea la care se referă aceste drepturi. Dacă o astfel de imagine este furată și depusă pe o pagină Web, proprietarul de drept poate dezvălui mesajul steganografic în cursul unui proces pentru a proba a cui este imaginea. Această tehnică este cunoscută sub denumirea de **filigranare** (eng.: *watermarking*). Ea este discutată în (Piva et. al, 2002).

Pentru mai multe despre steganografie, vedeți (Artz, 2001; Johnson și Jajoda, 1998; Katzenbeisser și Petitcolas, 2000 și Wainer. 2002).

8.10.3 Dreptul de autor

Confidențialitatea și cenzura sunt doar două din zonele unde tehnologia întâlnește politicul. O a treia este problema **dreptului de autor** (eng.: *copyright*). Acordarea dreptului de autor înseamnă a garanta creatorului de **PI** (*proprietate intelectuală* – eng.: *Intellectual Propriety*), inclusiv scriitori, artiști, muzicieni, fotografi, cineaști, coreografi și alții, dreptul exclusiv de a exploata proprietatea lor intelectuală pentru o perioadă de timp, de obicei viața autorului plus 50 sau 75 de ani în cazul proprietății comune. După ce expiră dreptul de autor asupra unei lucrări, aceasta trece în domeniul public și oricine o poate folosi sau vinde după bunul plac. De exemplu, proiectul Gutenberg (*www.promo.net/pg*), de exemplu, a pus pe Web mii de lucrări din domeniul public (de exemplu, Shakespeare, Twain, Dickens). În anul 1998, Congresul Statelor Unite a extins dreptul de autor la încă 20 de ani la cererea Hollywood-ului, care a motivat cererea prin faptul că fără o extindere a termenului, nimeni nu ar mai crea nimic, niciodată. Prin contrast, patentele durează doar 20 de ani și totuși oamenii încă mai inventează lucruri.

Dreptul de autor a ajuns în prim-plan când Napster, un serviciu de traficare a muzicii, a atins 50 de milioane de membri. Deși Napster nu copia muzica, tribunalele au decis că, deoarece deținea o bază de date ce reflecta cine avea ce cântec, a contribuit la încălcarea legii, cu alte cuvinte a ajutat pe alții să încălcat legea. Deși nimeni nu pretinde că dreptul de autor este o idee proastă (mulți pretind

că termenul este mult prea lung, favorizând marile corporații față de public), noua generație de partajare a muzicii are deja implicații etice serioase.

De exemplu, să considerăm o rețea de la egal la egal în care oamenii partajează fișiere legitime (muzică din domeniul public, filme video, tratate religioase ce nu sunt secrete înregistrate etc.) și câteva cu drepturi de autor. Să presupunem că toți sunt permanent conectați prin ADSL sau cablu. Fiecare calculator are un index cu ce este pe discul dur plus o listă a altor membri. Cineva care caută un anumit obiect poate să aleagă aleator un membru și să vadă dacă obiectul căutat este în posesia lui. Dacă nu, poate verifica toți membrii din lista acelei persoane și toți membri din listele lor ș.a.m.d. Calculatoarele sunt foarte bune la astfel de lucruri. Dacă a găsit elementul respectiv, solicitantul doar îl copiază.

Dacă acel obiect este cu drept de autor, există șanse ca solicitantul să încalce legea (deși pentru transferuri internaționale, nu este clar ce legi sunt aplicabile). Dar ce putem spune despre furnizor? Este un delict să păstrezi muzică pentru care ai plătit și ai descărcat-o legal pe discul tău dur unde alții o pot găsi? Dacă aveți o cabană neîncuiată la țară și un hoț de PI se furișează purtând un calculator portabil și un scanner, copiază o carte ce are drepturi de autor și apoi pleacă, ești *tu* vinovat de delictul de a nu fi protejat dreptul de autor al altcuiva?

Dar este mai mult de discutat pe tema dreptului de autor. Există o mare dispută între Hollywood și industria calculatoarelor. Primul dorește protecție la sânge a întregii proprietăți intelectuale și cel de-al doilea nu vrea să fie polițistul Hollywood-ului. În octombrie 1998, congresul a votat **Legea dreptului de autor digital** (eng.: *Digital Millennium Copyright Act - DMCA*) care stipulează că este delict să treci peste orice mecanism de protecție a unei lucrări cu drept de autor sau să spui altora cum să o facă. O legislație similară este pusă în practică și în Uniunea Europeană. În timp ce nimeni nu crede că piraților din extremul Orient ar trebui să le fie permisă copierea lucrărilor cu drept de autor, mulți oameni cred că DMCA schimbă total balanța între interesul proprietarului de drept de autor și interesul public.

Un exemplu! În septembrie 2000, un consorțiu al industriei muzicale însărcinat să realizeze un sistem infailibil pentru a vinde de muzică online a sponsorizat un concurs, invitând oamenii să încerce să spargă sistemul (care este chiar lucrul cel mai potrivit pe care să-l faci cu un sistem de securitate nou). O echipă de cercetători în securitate de la diverse universități, condusă de profesorul Edward Felten din Princeton, a primit provocarea și a spart sistemul. Apoi au scris o lucrare despre ce au găsit ei și au trimis-o unei conferințe USENIX de securitate, unde a fost recenzată și acceptată. Înainte de prezentarea lucrării, Felten a primit o scrisoare de la Asociația Industriei Americane de Înregistrări (eng.: *Recording Industry Association of America*) ce amenința autorii cu un proces ce avea la bază DMCA dacă ar fi publicat lucrarea.

Răspunsul lor a fost intentarea unui proces, rugând un tribunal federal să decidă dacă este încă legal să publici lucrări științifice despre securitate. Cu spaima unei decizii definitive împotriva lor, industria și-a retras plângerea și tribunalul a casat procesul lui Felten. Fără îndoială că industria a fost motivată de slăbiciunea cauzei lor: ei au invitat oamenii să încerce să le spargă sistemul și apoi tot ei au amenințat cu darea în judecată pe câțiva care au acceptat provocarea. Cu amenințarea retrasă, lucrarea a fost publicată (Craver et. al, 2001). O nouă confruntare este inevitabilă.

O problemă apropiată este răspândirea doctrinei de utilizare corectă (eng.: *fair use doctrine*), ce a fost stabilită prin hotărâri judecătorești în diverse țări. Această doctrină spune că cumpărătorii lucrării cu drept de autor au câteva drepturi limitate de a copia lucrarea, inclusiv dreptul de a cita părți din ea pentru scopuri științifice, de a o folosi în scopuri didactice în școli și licee și în unele cazuri de a face copii de rezervă pentru folosul personal, în cazul în care mediul original nu mai este bun. Tes-

tele pentru ceea ce constituie o utilizare corectă includ (1) dacă materialul este comercial, (2) ce procentaj din întreg este copiat și (3) efectul copierii asupra vânzării lucrării. Din moment ce DMCA și legi similare din Uniunea Europeană întârzie eludarea schemelor de protecție, aceste legi interzic de asemenea o utilizare corectă. De fapt, DMCA ia utilizatorilor niște drepturi istorice pentru a da mai multă putere vânzătorilor de conținut. O confruntare finală este inevitabilă.

O altă evoluție a lucrurilor ce restrânge chiar și DMCA-ul în schimbarea balanței între proprietarii dreptului de autor și utilizatori este **TCPA** (eng.: *Trusted Computing Platform Alliance* – rom.: *Alianța platformelor de calcul de încredere*) condusă de Intel și Microsoft. Ideea este de a face ca microprocesorul și sistemul de operare să monitorizeze atent comportamentul utilizatorului în diverse moduri (de exemplu, când ascultă muzică pirat) și să prevină comportamentele nedorite. Sistemul permite chiar deținătorilor de conținut să manipuleze la distanță PC-ul utilizatorului pentru a schimba regulile când aceasta este neapărat necesar. Nu trebuie să mai spunem că urmările sociale ale acestei scheme sunt imense. Este nostim că în final industria are grijă de securitate, dar este lamentabil că întreaga atenție este concentrată asupra aplicării legii dreptului de autor și nu asupra virușilor, spărgătorilor, intrușilor și altor probleme de securitate de care sunt interesate cele mai multe persoane.

Pe scurt, legiuitorii și avocații vor fi ocupați în anii următori să pună echilibreze interesele economice ale proprietarilor de drepturi de autor și interesul public. Cyber-spațiul nu este diferit față de piața cărnii: în mod constant ea asmute un grup asupra celuilalt, de aici rezultând lupte pentru putere, litigii și (sperăm) în final o urmă de reconciliere, cel puțin până va apare o nouă tehnologie diversionistă.

8.11 REZUMAT

Criptografia este o unealtă ce poate fi folosită pentru menținerea confidențialității informației și pentru a asigura integritatea și autenticitatea sa. Toate sistemele criptografice moderne sunt bazate pe principiul lui Kerckhoff de avea un algoritm cunoscut public și o cheie secretă. Mulți algoritmi de criptografici folosesc transformări complexe, substituții și permutări pentru a transforma textul simplu în text cifrat. Totuși, dacă criptografia cuantică poate fi pusă în practică, folosirea unor chei acoperitoare poate fi într-adevăr un sistem criptografic imposibil de spart.

Algoritmii criptografici pot fi divizați în algoritmi cu chei simetrice și cu chei publice. Algoritmii cu chei simetrice amestecă biții într-o serie de runde parametrizate de cheie pentru a schimba textul simplu în text cifrat. DES triplu și Rijndael (AES) sunt cei mai cunoscuți algoritmi cu cheie simetrică în momentul de față.

Algoritmii cu chei publice au proprietatea că sunt folosite diferite chei pentru criptare și decriptare și că cheia de decriptare nu poate fi derivată din cheia de criptare. Aceste proprietăți fac posibilă publicarea cheii publice. Principalul algoritm cu cheie publică este RSA, care își trage puterea din faptul că este foarte dificil să se factorizeze numerele mari.

Documentele legale, comerciale și altele au nevoie de semnături. Ca urmare, au fost propuse diverse scheme pentru semnăturile digitale, folosind algoritmi cu chei simetrice și cu chei publice. Uzual, mesajele ce trebuie semnate sunt rezumate folosind algoritmi precum MD5 și SHA-1 și apoi sunt semnate rezumatele și nu mesajele originale.

Gestiunea cheilor publice poate fi făcută folosind certificate, care sunt documente ce leagă un principal de o cheie publică. Certificatele sunt semnate de către o autoritate de încredere sau de către cineva aprobat (recursiv) de o autoritate de încredere. Rădăcina lanțului trebuie obținută înainte, dar programele de navigare au incluse multe certificate de rădăcini în ele.

Aceste instrumente criptografice pot fi folosite pentru a securiza traficul în rețea. IPsec operează la nivelul rețea, criptând fluxurile de pachete de la o gazdă la alta. Zidurile de protecție pot verifica traficul ce iese sau intră într-o organizație, deseori pe baza protocolului și a portului folosit. Rețelele private virtuale pot simula o veche rețea închiriată pentru a asigura proprietățile necesare de securitate. În final, rețelele fără fir au nevoie de o bună securitate pe care WEP 802.11 nu o aduce, deși 802.11i ar trebui să îmbunătățească lucrurile destul de mult.

Când două părți stabilesc o sesiune, trebuie să se autentifice una către alta și dacă este nevoie să stabilească o cheie comună a sesiunii. Există diverse protocoale de autentificare, inclusiv câteva ce folosesc o terță parte verificată, Diffie-Hellman, Kerberos și criptografia cu cheie publică.

Securitatea mesajelor electronice poate fi obținută printr-o combinație de tehnici pe care le-am studiat în cadrul acestui capitol. De exemplu, PGP comprimă mesajul, apoi îl criptează folosind IDEA. El trimite cheia IDEA criptată cu cheia publică a receptorului. În plus, rezumă mesajul și trimite rezumatul semnat pentru verificarea integrității mesajului.

Securitatea pe Web este de asemenea un subiect important, începând cu securitatea numelor. DNSsec oferă o cale pentru prevenirea păcălirii DNS-ului, așa cum fac și numele cu auto-certificare. Majoritatea paginilor Web de comerț electronic folosesc SSL pentru a stabili sesiuni securizate, autentificate între client și server. Diverse tehnici sunt folosite pentru a rezolva probleme legate de codul mobil, în special depunerea într-un mediu protejat și semnarea codului.

Internet-ul ridică multe probleme în care tehnologia interacționează puternic cu politica publică. Câteva dintre zone includ confidențialitatea, libertatea de exprimare și dreptul de autor.

8.12 PROBLEME

1. Spargeți următorul cifru monoalfabetic. Textul în clar, constând numai din litere, este un fragment dintr-o poezie a lui Lewis Carroll.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutyhc fuk zhx pfdkfdi ntcm fzld pthcm
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Spargeți următorul cifru de transpoziție pe coloane. Textul este luat dintr-o carte despre calculatoare, astfel încât „computer” este un cuvânt probabil. Textul constă numai din litere (fără spații). Pentru claritate, textul cifrat este împărțit în blocuri de 5 caractere.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lborl otoex

3. Găsiți o cheie acoperitoare de 77 de biți ce generează textul „Donald Duck” din textul cifrat din fig. 8-4.

4. Criptografia cuantică necesită un tun fotonic ce poate, la cerere, să tragă un singur foton ce conține un singur bit. În această problemă, calculați câți fotoni cară un bit pe o legătură de fibră de 100 Gbps. Presupuneți că lungimea unui foton este egală cu lungimea lui de undă, care pentru scopul acestei probleme este de 1 micron. Viteza luminii în fibră este de 20 cm/ns.

5. Dacă Trudy capturează și regenerează fotonii când se folosește criptografia cuantică, ea va amesteca o parte din ei și va provoca erori în cheia acoperitoare a lui Bob. Ce fracțiune din biții din cheia acoperitoare a lui Bob vor fi eronați, în medie?

6. Un principiu criptografic fundamental spune că toate mesajele trebuie să aibă redundanță. Dar de asemenea știm că redundanța ajută un intrus să-și verifice cheia ghicită. Considerați două forme de redundanță. În primul caz, cei n biți inițiali ai textului simplu conțin un tipar cunoscut. În al doilea, n biți de al sfârșitul mesajului conțin un rezumat a mesajului. Din punct de vedere al securității, sunt acestea două echivalente? Discutați răspunsul dumneavoastră.

7. În fig. 8-6, blocurile P și S alternează. Deși acest aranjament este agreabil, nu este mai sigur să avem mai întâi toate blocurile P și apoi toate blocurile S?

8. Proiectați un atac la DES bazat pe faptul că textul constă numai din litere mari ASCII, plus spațiu, virgulă, punct și virgulă, întoarcere la capăt (CR) și linie nouă (LF). Nu se știe nimic despre biții de paritate ai textului.

9. În text am calculat că o mașină de spart cifruri cu un miliard de procesoare ce ar putea analiza o cheie într-o picosecundă ar lua numai 10^{10} ani pentru a sparge versiunea AES de 128 de biți. Totuși, mașinile curente ar putea avea 1024 de procesoare și le-ar trebui 1 ms pentru a analiza o cheie, așa că ne trebuie un factor de îmbunătățire a performanței de 10^{15} pentru a obține mașina ce sparge AES. Dacă legea lui Moore (puterea de procesare se dublează la fiecare 18 luni) se menține în vigoare, de câți ani este nevoie pentru a putea construi mașina?

10. AES suportă chei de 256 de biți. Câte chei are AES-256? Verificați dacă găsiți numere în fizică, chimie, astronomie care au aceeași dimensiune. Folosiți Internet-ul pentru a vă facilita căutarea de numere mari. Trageți o concluzie din cercetarea dumneavoastră.

11. Să presupunem că un mesaj a fost criptat folosind DES cu înlănțuirea blocurilor cifrate. Un bit al textului cifrat în blocul C_i este transformat accidental din 0 în 1 în timpul transmisiei. Cât de mult text va fi deformat ca urmare a acestui fapt?

12. Să considerăm din nou înlănțuirea blocurilor cifrate. În loc să transformăm un bit 0 în 1, un bit 0 este inserat în fluxul textului cifrat după blocul C_i. Cât de mult text va fi deformat?

13. Comparați înlănțuirea blocurilor cifrate cu modul cu reacție cifrată în funcție de numărul de operații de criptare folosite pentru transmiterea unui fișier mare. Care este mai eficient și cu cât?

14. Folosind sistemul de criptare cu chei publice RSA cu a=1, b=2 etc.,

dacă $p=7$ și $q=11$, listați 5 valori permise pentru d.
Dacă $p=13$, $q=31$ și $d=7$, cât este e?
Folosind $p=5$, $q=11$ și $d=27$, găsiți valoarea lui e și criptați „abcdefghij".

15. Să presupunem că un utilizator, Maria, descoperă că cheia sa privată RSA ($d1$, $n1$) este aceeași cu cheia publică RSA ($e2$, $n2$) a altui utilizator, Frances. Cu alte cuvinte, $d1 = e2$ și $n1 = n2$. Ar trebui să se gândească să-și schimbe cheile sale publică și privată? Explicați răspunsul dumneavoastră.

16. **Să considerăm** folosirea modului contor, cum este arătat în fig. 8-15, dar cu $IV = 0$. Folosirea lui 0 amenință securitatea cifrului în general?

17. Protocolul de semnătură din fig. 8-18 are următoarele puncte slabe. Dacă Bob se defectează, el poate pierde conținutul RAM-ului propriu. Ce probleme pot apărea și cum pot fi prevenite?

18. În fig. 8-20, putem vedea cum Alice poate trimite lui Bob un mesaj semnat. Dacă Trudy înlocuiește P, Bob poate să-l detecteze. Dar ce se întâmplă dacă Trudy înlocuiește ambii P și semnătura?

19. Semnăturile digitale au o slăbiciune potențială datorită utilizatorilor leneși. În tranzacțiile din comerțul electronic, un contract poate fi redactat și utilizatorului i se cere să semneze dispersia sa SHA-1. Dacă utilizatorul nu verifică dacă contractul și dispersia corespund, utilizatorul ar putea să semneze din neatenție un alt contract. Să presupunem că Mafia încearcă să exploateze această slăbiciune pentru a câștiga ceva bani. Vor face o pagina Web cu plată (de exemplu, pornografie, jocuri de noroc, etc.) și va cere noilor clienți numărul cărții de credit. Apoi, ei trimit un contract în care spun că clientul dorește să folosească serviciile lor și să plătească cu carte de credit și roagă clientul să semneze, știind că cei mai mulți vor semna fără să verifice dacă contractul și rezumatul corespund. Arătați cum poate Mafia să cumpere diamante de la bijutier legitim pe Internet și acestea să fie plătite de către clienții nesuspicioși.

20. O clasă de matematică are 20 de studenți. Care este probabilitatea ca cel puțin doi studenți să aibă aceeași zi de naștere? Presupuneți că nimeni nu s-a născut pe 29 februarie, astfel că sunt 365 de zile de naștere posibile.

21. După ce Ellen s-a confesat lui Marilyn că a păcălit-o în ceea ce-l privește pe Torn, Marilyn a reușit să evite această problemă dictând mesajele ulterioare unei mașini de dictat, și punând noua secretară să le introducă. Marilyn a plănuit să examineze mesajele de pe terminalul ei după ce au fost introduse pentru a fi sigură că acestea conțin propriile cuvinte. Poate noua secretară să folosească atacul zilei de naștere, ca să falsifice un mesaj și dacă da, cum? *Indicație*: Poate.

22. Considerați încercarea nereușită a lui Alice de a obține cheia publică a lui Bob în fig. 8-23. Presupuneți că Bob și Alice au în comun o cheie secretă, dar Alice tot vrea cheia publică a lui Bob. Există un mod prin care ea poate fi obținută într-un mod securizat? Dacă da, cum?

23. Alice dorește să comunice cu Bob, folosind criptografia cu chei publice. Ea stabilește o conexiune cu cineva sperând că acesta este Bob. Ea îi cere cheia lui publică și el i-o trimite în text clar

împreună cu un certificat X.509 semnat de către rădăcina CA. Alice are deja cheia publică a rădăcinii CA. Ce pași trebuie să îndeplinească Alice pentru a vedea că vorbește cu Bob? Presupuneți că lui Bob nu-i pasă cu cine vorbește (de exemplu, Bob este un fel de serviciu public).

24. Presupuneți că un sistem folosește PKI bazată pe o ierarhie cu structură de arbore de CA-uri. Alice dorește să comunice cu Bob și primește un certificat de la Bob semnat de CA *X* după ce s-a stabilit un canal de comunicație cu Bob. Să considerăm că Alice nu a auzit niciodată de *X*. Ce pași trebuie urmați de Alice pentru a verifica dacă vorbește cu Bob ?

25. Se poate ca IPsec cu AH să fie folosit în modul transport de una dintre mașinile ce se află în spatele unei cutii NAT ? Explicați răspunsul.

26. Precizați un avantaj al folosirii HMAC față de folosirea RSA pentru semnarea rezumatelor SHA-1.

27. Dați un motiv pentru care un zid de protecție ar putea fi configurat să cerceteze traficul dinspre exterior. Dați un motiv pentru care ar putea fi configurat să inspecteze traficul spre exterior. Credeți că aceste inspecții au șanse de succes ?

28. Formatul pachetului WEP este arătat în fig. 8-31. Să presupunem că suma de control este de 32 de biți, calculată prin SAU ECLUSIV asupra tuturor cuvintelor de 32 de biți din încărcătura utilă luate împreună. De asemenea să presupunem că problemele cu RC4 sunt corectate prin înlocuirea cu un cifru-șir ce nu are slăbiciuni și că toate elementele IV sunt extinse la 128 de biți. Există vreo cale pentru ca un intrus să spioneze sau să interfereze cu traficul fără a fi detectat?

29. Să presupunem că o organizație ce folosește VPN pentru a conecta securizat siturile sale din Internet. Este nevoie ca un utilizator din această organizație să folosească criptarea sau alt mecanism de securitate pentru a comunica cu alt utilizator din cadrul organizației?

30. Modificați un mesaj în protocolul din fig. 8-34 pentru a-l face rezistent la atacurile prin reluare. Explicați de ce funcționează modificarea.

31. Schimbul de chei Diffie-Hellman este folosită pentru stabilirea unei chei secrete între Alice și Bob. Alice trimite lui Bob (719, 3, 191). Bob răspunde cu (543). Numărul secret al lui Alice este x = 16. Care este cheia secretă?

32. Dacă Alice și Bob nu s-au întâlnit niciodată, nu au secrete comune, și nu au certificate, ei totuși pot să stabilească o cheie secretă comună folosind algoritmul Diffie-Hellman. Explicați de ce le este foarte greu să se apere împotriva unui atac omul-din-mijloc.

33. În protocolul din fig. 8-39, de ce este A trimis în clar împreună cu cheia de sesiune criptată?

34. În protocolul din fig. 8-39, am arătat că a începe fiecare mesaj transmis în clar cu 32 de biți zero este riscant. Să presupunem că fiecare mesaj începe cu un număr aleatoriu al utilizatorului, care este practic o a doua cheie secretă cunoscută doar de utilizatorul ei și de KDC. Se elimină în acest mod atacul textului clar cunoscut? De ce?

35. În protocolul Needham-Schroeder, Alice generează 2 provocări, R_A și R_{A2}. Aceasta seamănă cu o exagerare. Nu ar fi fost suficientă una singură?

36. Să presupunem că o organizație folosește Kerberos pentru autentificare. Ce se întâmplă, din punctul de vedere al securității și al disponibilității serviciului, dacă AS sau TGS cad?

37. În protocolul de autentificare cu chei publice din fig. 8-43, în mesajul 7, R_B este criptat cu K_S. Este această criptare necesară, sau ar fi fost potrivit să fie trimisă ca text clar? Explicați răspunsul.

38. Terminalele din punctele de vânzare care folosesc cartele cu bandă magnetică și coduri PIN au un defect fatal: un negustor răuvoitor poate modifica cititorul de coduri propriu pentru a captura și a memora toată informația de pe cartelă precum și codul PIN pentru a transmite ulterior tranzacții suplimentare (falsificate). Următoarea generație de terminale din punctele de vânzare va folosi cartele cu unități centrale complete, tastatură și monitor pe cartelă. Implementați un protocol pentru acest sistem, pe care negustorii răuvoitori să nu-l poată sparge.

39. Enumerați două motive pentru care PGP comprimă mesajele.

40. Presupunând că toată lumea de pe Internet folosește PGP, ar putea un mesaj PGP să fie trimis la o adresă Internet arbitrară și să fie decodificat corect de toți cei implicați? Discutați răspunsul.

41. Atacului prezentat în fig. 8-47 îi lipsește un pas. Acest pas nu este necesar pentru ca atacul să funcționeze, dar includerea lui ar putea reduce eventualele suspiciuni ulterioare. Care este pasul lipsă?

42. S-a propus împiedicarea păcălirii DNS-ului folosind predicția identificatorilor, metodă în care serverul pune un identificator aleatoriu în loc să utilizeze un contor. Discutați aspectele legate de securitate ale acestei abordări.

43. Protocolul de transport de date al SSL implică două numere ad-hoc și o cheie primară. Ce rol are folosirea numerelor ad-hoc (dacă are vreunul)?

44. Imaginea fin fig. 8-55(b) conține textul ASCII a cinci piese de Shakespeare. Ar fi posibil să fie ascunsă muzică printre zebre în loc de text? Dacă da, cum ar funcționa și cât de mult ați putea ascunde în acea imagine? Dacă nu, de ce nu?

45. Alice era o utilizatoare fidelă a unui retransmițător anonim de tip 1. Ea trimitea multe mesaje grupului ei de știri preferat, *alt.fanclub.alice* și toată lumea știa că acestea vin de la Alice pentru că toate purtau același pseudonim. Presupunând că retransmițătorul funcționa corect, Trudy nu putea să pretindă că e Alice. După ce toate retransmițătoarele de tip 1 au fost desființate, Alice s-a mutat la unul care utilizează criptografia și a început o nouă serie de mesaje în grupul ei. Imaginați-vă un mod de a o opri pe Trudy să trimită mesaje noi grupului, pretinzând că este Alice.

46. Căutați pe Internet nu caz interesant implicând confidențialitatea și scrieți o prezentare de o pagină despre acesta.

47. Căutați pe Internet un caz ajuns la tribunal, implicând dreptul de autor și utilizarea corectă și scrieți o prezentare de o pagină, rezumând ceea ce ați găsit.

48. Scrieți un program care criptează datele de intrare, aplicându-le operația XOR cu un șir-cheie. Găsiți sau scrieți și un generator de numere aleatorii pentru a putea genera șirul-cheie. Programul ar trebui să se comporte ca un filtru, preluând text clar de la intrarea standard și producând

text cifrat la ieșirea standard (și invers). Programul trebuie să aibă un singur parametru, cheia de la care pornește generatorul de numere aleatoare.

49. Scrieți o procedură care calculează rezumatul unui bloc de date folosind SHA-1. Procedura trebuie să aibă doi parametri: o referință la zona tampon de intrare și o referință la o zonă tampon, de 20 de octeți, de ieșire. Pentru a vedea specificațiile exacte ale SHA-1, căutați pe Internet FIPS 180-1, care este specificația completă.

9

RECOMANDĂRI DE LECTURĂ ȘI BIBLIOGRAFIE

Aici se încheie studiul nostru despre rețelele de calculatoare, dar această lucrare este doar un început. Multe subiecte interesante nu au fost tratate la nivelul de detaliu pe care l-ar fi meritat, iar altele au fost omise în totalitate, din lipsă de spațiu. Pentru cititorii care doresc să continue studiul rețelelor de calculatoare furnizăm în acest capitol câteva sugestii de lecturi posibile, precum și o listă bibliografică.

9.1 SUGESTII PENTRU LECTURI VIITOARE

Există o literatură vastă despre toate aspectele rețelelor de calculatoare. Trei reviste, care publică frecvent articole în acest domeniu sunt *IEEE Transactions on Communications, IEEE Journal on Selected Areas in Communications* și *Computer Communication Review* . De asemenea, multe alte reviste publică ocazional articole cu acest subiect.

IEEE mai publică și trei reviste ilustrate - *IEEE Internet Computing*, *IEEE Network Magazine* și *IEEE Communications Magazine* - care conțin studii, îndrumări practice, studii de caz despre rețele. Primele două reviste se referă cu precădere la arhitectură, standarde și software, iar ultima se orientează către tehnologia comunicațiilor (fibră optică, sateliți și așa mai departe).

În plus, există un număr de conferințe anuale sau bianuale care atrag multe lucrări despre rețele și sisteme distribuite, în particular *SIGCOMM Annual Conference, The International Conference on Distributed Computer Systems* și *The Symposium on Operating Systems Principles*

Enumerăm în continuare, în ordinea capitolelor cărții, câteva sugestii de lecturi suplimentare. Cele mai multe dintre acestea sunt îndrumări practice sau studii asupra subiectelor la care se referă. Câteva dintre ele sunt capitole din diverse manuale.

9.1.1 Lucrări introductive și generale

Bi et. al, „Wireless Mobile Communications at the Start of the 21st Century"

Un nou secol, o nouă tehnologie – un subiect incitant. După un istoric al comunicațiilor fără fir, sunt tratate aspectele de bază, inclusiv standarde, aplicații, Internet și tehnologie.

Comer, *The Internet Book*

Cine caută o introducere simplă în Internet ar trebui să citească această lucrare. Comer descrie istoria, dezvoltarea, tehnologia, protocoalele și serviciile Internet-ului în termeni pe care novicii îi pot înțelege, dar datorită cantității de material acoperite, cartea sa prezintă interes și pentru cititorii mai avansați.

Garber, „Will 3G Really Be the Next Big Wireless Technology?"

De la telefoanele mobile din a treia generatie se așteaptă să fie capabile de transmisii de voce și de date la rate de transfer de până la 2 Mbps, însă startul acestei tehnologii a fost unul lent. Promisiunile, obstacolele, tehnologia, considerentele politice și economice ale utilizării comunicației wireless în bandă largă sunt tratate în acest articol ușor de citit.

IEEE Internet Computing, Ianuarie – Februarie 2000

Primul număr al revistei *IEEE Internet Computing* din noul mileniu a apărut exact în formatul pe care l-am fi așteptat: personalități care în mileniul precedent au contribuit la crearea Internetului au fost invitate să-și expună viziunile asupra evoluției Internetului in mileniul următor. Printre experți se numără Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy. Pentru a valorifica la maxim acest material, este recomandat să așteptați 500 de ani și *apoi* să citiți profețiile lor.

Kipnis, „Beating the System: Abuses of the Standards Adoption Process"

Comisiile care se ocupă cu adoptarea standardelor încearcă să mențină o poziție neutră în demersurile lor, însă din nefericire există companii care încearcă să abuzeze de sistem. De exemplu, s-au înregistrat în repetate rânduri cazuri în care după adoptarea unui standard, o anumită companie care a participat la dezvoltarea acestuia să anunțe ca deține un brevet ce stă la baza standardului și să acorde licențe doar anumitor companii agreate și la prețuri stabilite doar după criterii proprii. Articolul este un început excelent în studiul „părții întunecate" a procedurilor de adoptare a standardelor.

Kyas și Crawford, *ATM Networks*

ATM, cotat la un moment dat ca protocolul de rețea al viitorului, joacă încă un rol important în cadrul sistemului telefonic. Această lucrare se constituie într-un ghid la zi, conținând informații detaliate despre protocoalele ATM și despre modul de integrare al acestora cu rețele bazate pe protocolul IP.

Kwok, „A Vision for Residential Broadband Service”

Dacă doriți să aflați viziunea Microsoft asupra serviciului de video la cerere în anul 1995, citiți acest articol. Cinci ani mai târziu această viziune era complet depășită. Valoarea acestui articol constă în a demonstra că până și persoane puternic motivate și cu un nivel înalt de cunoștințe în domeniu se găsesc în imposibilitatea de a estima cu precizie evoluția chiar și pe un termen de numai 5 ani. Ar trebui să fie o lecție pentru noi toți.

Naughton, *A Brief History of the Future*

De fapt, cine este inventatorul Internetului? Multe persoane își asumă acest merit, și pe bună dreptate, din moment ce mulți au contribuit în diferite moduri la crearea sa. Acest scurt istoric al Internetului arată cum s-au întâmplat lucrurile într-o manieră spirituală și plăcută, presărată cu anecdote, cum ar fi convingerea AT&T, exprimată în mod repetat, că nu există nici un viitor în domeniul comunicațiilor digitale.

Perkins, „Mobile Networking in the Internet”

Pentru a vă forma o imagine de ansamblu asupra protocoalelor din domeniul rețelelor mobile, nivel cu nivel, aceasta este lucrarea pe care trebuie să o consultați. Sunt examinate rând pe rând nivelurile de la fizic până la transport, precum și middleware-ul, considerente de securitate și rețelele ad-hoc.

Teger și Waks, „End-User Perspectives on Home Networking”

Rețelele pentru utilizatori individuali nu se aseamănă cu cele pentru corporații. Aplicațiile sunt diferite (preponderent multimedia), echipamentele provin de la o gamă largă de producători iar utilizatorii au un nivel scăzut de pregătire tehnică și o toleranță mică la defecțiuni de funcționare. Citiți această carte pentru a afla mai multe.

Varshney și Vetter, „Emerging Mobile și Wireless Networks”

Acesta este o altă introducere în domeniul comunicațiilor fără fir. Sunt discutate LAN-urile fără fir, buclele locale fără fir, sateliții precum și unele aspecte legate de software-ul și aplicațiile aferente.

Wetteroth, *OSI Reference Model for Telecommunications*

Deși protocoalele stivei OSI nu mai sunt utilizate ca atare, modelul cu șapte niveluri a căpătat o mare popularitate. Lucrarea oferă mai multe explicații referitoare la modelul OSI și își propune să aplice acest model rețelelor de telecomunicații (în contrast cu rețelele de calculatoare), evidențiind unde se încadrează în stiva OSI diverse protocoale de telefonie și transmisie de voce.

9.1.2 Nivelul fizic

Abramson, „Internet Access Using VSATs”

Stațiile terestre de dimensiune redusă devin din ce în ce mai utilizate atât pentru sistemele de telefonie din mediul rural cât și pentru accesul Internet al corporațiilor din țările dezvoltate. Totuși, natura traficului diferă radical în cele două situații, așa că sunt necesare protocoale diferite. În acest articol, inventatorul protocolului ALOHA discută despre numeroase metode de alocare a benzii de transmisie utilizate în sisteme VSAT.

Alkhatib et. al, „Wireless Data Networks: Reaching the Extra Mile”

Această lucrare, concepută ca un îndrumar practic, este un bun punct de pornire pentru o introducere rapidă în terminologia și tehnologia utilizată de rețelele fără fir.

Azzam și Ransom, *Broadband Access Technologies*

Sistemul de telefonie, fibrele optice, ADSL, rețelele de televiziune prin cablu, sateliți, chiar și liniile de înaltă tensiune ca tehnologii de acces în rețea sunt discutate în această lucrare. Alte subiecte includ rețele pentru utilizatori individuali, servicii, performanțele rețelelor și standarde. Lucrarea se încheie cu biografiile marilor companii din industria telecomunicațiilor și rețelelor, dar luând în considerare evoluția rapidă a acestui domeniu, acest capitol va avea probabil o existență mai scurtă decât capitolele referitoare la tehnologii.

Bellamy, *Digital Telephony*

Această lucrare demnă de încredere conține tot ce ați dorit vreodată să știți despre sistemul telefonic și chiar mai mult. În particular, sunt interesante capitolele despre transmisie și multiplexare, comutare digitală, fibră optică, telefonie mobilă și DSL .

Berezdivin et. al, „Next-Generation Wireless Communications Concepts and Technologies"

Autorii sunt cu un pas înaintea tuturor celorlalți. „Următoarea generație" din titlu se referă la a patra generație de comunicații fără fir. De la aceste rețele se așteaptă să furnizeze servicii IP în orice loc, o legătură ireproșabilă la Internet cu lățime mare de bandă și o calitate excelentă a serviciilor. Aceste scopuri vor fi atinse prin alocarea inteligentă a spectrului, alocarea dinamică a resurselor și servicii adaptabile. Este o viziune relativ îndepărtată, dar același lucru se putea spune în 1995 și despre telefonia mobilă .

Dutta-Roy, „An Overview of Cable Modem Technology and Market Perspectives"

Televiziunea prin cablu a evoluat de la simplul sistem CATV la sisteme complexe de distribuție pentru televiziune, Internet și telefonie. Aceste schimbări au afectat în mare măsură și infrastructura de cablu. Acest articol merită citit pentru o dezbatere asupra fabricilor de cablu, a standardelor, a marketing-ului cu accentul căzând pe DOCSIS .

Farserotu și Prasad, „A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends"

O gamă largă de sateliți de comunicații de date se află pe orbită sau pe planșele de proiectare, incluzând Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic și iSk. Aceștia utilizează diverse tehnici printre care conductă curbă și comutare prin sateliți. Pentru o imagine de ansamblu a diferitelor tehnici și sisteme de sateliți această lucrare este un bun punct de pornire.

Hu și Li, „Satellite-Based Internet: A Tutorial"

Accesul la Internet prin satelit diferă de accesul prin liniile terestre. Nu numai că apare o problemă legată de întârziere, dar și rutarea și comutarea sunt diferite. În această lucrare, autorii dezbat câteva dintre problemele legate de accesul Internet prin intermediul sateliților.

Joel, „Telecommunications and the IEEE Communications Society"

Acest articol oferă un istoric compact dar cuprinzător al telecomunicațiilor, începând cu telegraful și încheind cu standardul 802.11. Totodată, sunt tratate aspecte legate de radio, telefonie, comutare analogică și digitală, cabluri submarine, transmisii digitale, ATM, stații de televiziune, televiziune prin cablu, comunicații prin fibră optică, telefonie mobilă, comutare de pachete, ARPANET și Internet.

Metcalfe, „Computer/Network Interface Design Lessons from Arpanet & Ethernet"

Deși inginerii au construit interfețe de rețea începând în urmă cu zeci de ani, unii se întreabă încă dacă au învățat ceva din această experiență. În acest articol, proiectantul Ethernet-ului spune cum se

construiește o interfață de rețea și ce se face cu ea odată ce a fost construită. El recunoaște sincer ce a făcut greșit și ce a făcut corect.

Palais, Fiber Optic Communication, ediția a 3-a

Deși cărțile despre tehnologia fibrei optice devin din ce în ce mai specializate, această lucrare este mai accesibilă decât multe altele. Ea acoperă ghidurile de undă, sursele de lumină, detectoarele de lumină, cuploarele, modularea, zgomotul și multe alte subiecte.

Pandya, „Emerging Mobile and Personal Communications Systems"

Acest articol este foarte potrivit ca o introducere scurtă și plăcută în sistemele de comunicație portabile personale. Una din cele nouă pagini conține o listă de 70 de acronime folosite în celelalte opt pagini.

Sarikaya, „Packet Mode in Wireless Networks: Overview of Transition of Third Generation"

Conceptul de bază a rețelelor celulare din generația a treia constă în transmisii de date fără fir. Acest articol este foarte potrivit pentru a obține o imagine de ansamblu asupra transmisiilor de date în rețelele din generația a doua și cum vor evolua acestea în rețelele din generația a treia. Sunt tratate subiecte ca GPRS, IS-95B, WCDMA si CDMA 2000.

9.1.3 Nivelul legătură de date

Carlson, PPP Design, Implementation and Debugging, editia a 2-a

Dacă vă interesează informații detaliate despre toate protocoalele ce formează suita PPP, inclusiv CCP (compresie) si ECP (criptare), acestă lucrare este de referință. Se pune un accent deosebit pe ANU PPP-2.3, o implementare larg răspândită a PPAG.

Gravano, *Introduction to Error Control Codes*

Erori se strecoară în aproape toate formele de comunicație digitală; din acest motiv au fost dezvoltate multe tipuri de coduri detectoare si corectoare de erori. Această lucrare descrie unele dintre cele mai importante coduri, de la de la Codul Hammimg până la codul Galois și codurile convoluționale, necesitând un volum minim de cunoștințe de algebră din partea cititorului.

Holzmann, Design and Validation of Computer Protocols

Cititorii interesați în aspectele mai formale ale protocoalelor legăturii de date (și similare) ar trebui să vadă această carte. Sunt prezentate aici specificarea, modelarea, corectitudinea și testarea acestor protocoale.

Peterson și Davie, *Computer Networks: A System Approach*

Capitolul 2 tratează multe aspecte legate de nivelul legatură de date, inclusiv încadrarea (framing), protocoalele start-stop, protocoale cu fereastră glisantă și LAN-urile IEEE 802.

Stallings, *Data and Computer Communications*

Capitolul 7 tratează aspecte legate de nivelul legatură de date cum ar fi controlul fluxului, detectarea erorilor și protocoale simple de nivel legatură de date, inclusiv start-stop și protocolul cu revenire cu n pași. Sunt discutate și protocoalele de tip HDLC.

9.1.4 Subnivelul de control al accesului la mediu

Bhagwat, „Bluetooth: Technology for Short-Range Wireless Apps”

Lucrarea reprezintă un bun punct de pornire pentru o introducere directă a sistemului Bluetooth. Sunt discutate protocoalele și profilurile de bază, radio, picorețele și conexiuni, urmate de o introducere a diverselor protocoale.

Bisdikian, „An Overview of the Bluetooth Wireless Technology”

La fel ca lucrarea precedentă a lui Bhagwat, și aceasta este un bun punct de pornire pentru a afla totul despre sistemul Bluetooth. Sunt prezentate printre altele picorețelele, stiva de protocoale și profilurile.

Crow et. al, „IEEE 802.11 Wireless Local Area Networks”

Această lucrare este un bun punct de pornire pentru o introducere simplă în tehnologia și protocoalele 802.11. Accentul cade pe subnivelul MAC. Sunt discutate atât controlul distribuit cât și controlul centralizat. În încheiere sunt prezentate câteva evaluări ale performanțelor 802.11 în urma simulărilor realizate în condiții diverse.

Eklund et. al, „IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access”

Standardul 802.16 adoptat de IEEE in 2002 presupune utilizarea comunicației fără fir pentru buclele locale, ceea ce ar revoluționa serviciile telefonice, aducând transmisia fără fir în bandă largă către locuințe. În această privire de ansamblu, autorii expun principalele aspecte tehnologice ce țin de acest standard.

Kapp, „802.11: Leaving the Wire Behind”

Această scurtă introducere a standardului 802.11 acoperă chestiunile de bază, protocoalele și standardele relevante.

Kleinrock, „On Some Principles of Nomadic Computing and Multi-Access Communications”

Partajarea canalului de transmisie in medii neghidate este o problemă mai complexă decat partajarea canalului de transmisie in medii ghidate. În acest articol sunt expuse probleme ca topologii dinamice, rutare, consumul de energie precum și alte aspecte legate de accesul la canalul de transmisie al echipamentelor mobile fără fir.

Miller și Cummins, *LAN Technologies Explained*

Aveți nevoie să cunoașteți cât mai multe despre tehnologiile care pot fi utilizate în LAN-uri? Această lucrare acoperă cele mai multe dintre aceste tehnologii, inclusiv FDDI și token ring, precum și larg răspânditul Ethernet. În timp ce primele două sunt rar utilizate în instalările noi de rețele, multe rețele deja existente le mai utilizează încă, iar rețelele cu topologie de inel sunt încă des întâlnite (de exemplu SONET).

Perlman, *Interconnections*, ediția a 2-a

Lucrarea reprezintă o expunere demnă de încredere, dar totuși distractivă, despre punți, rutere și rutare în general. Autorul este cel care a proiectat algoritmul “spanning tree” utilizat de punțile IEEE 802 și este una dintre autoritățile de nivel mondial în ceea ce privește diverse aspecte ale rețelelor.

Webb, „Broadband Fixed Wireless Access”

Această lucrare răspunde la întrebările „de ce” și „cum” în ceea ce privește transmisiile fără fir în bandă largă fixă. Secțiunea „de ce” argumentează că utilizatorii nu doresc adrese de e-mail separate pentru acasă și pentru serviciu, numere diferite de telefon pentru acasă, la serviciu sau pentru telefonul mobil, un cont pentru discuții (chat) și un număr sau două de fax. În schimb, ei preferă un sistem unic, integrat, care să funcționeze pretutindeni. Accentul în secțiunea referitoare la tehnologie cade asupra nivelului fizic și sunt discutate subiecte ca număr de purtătoare, comparații între TDD și FDD, între modulare fixă și modulare adaptivă.

9.1.5 Nivelul rețea

Bhatti si Crowcroft, „QoS Sensitive Flows: Issues in IP Packet Handling”

Una din modalitățile de a obține o mai bună calitate a serviciilor într-o rețea este planificarea riguroasă a plecării pachetelor din fiecare ruter. În această lucrare este prezentată o serie de algoritmi de planificare a pachetelor precum și alte detalii legate de acest subiect.

Chakrabarti, „QoS Issues in Ad Hoc Wireless Networks”

Rutarea în rețelele ad-hoc de calculatoare portabile care se află la distanță mică unele de altele este destul de dificilă și fără a ține cont de calitatea serviciilor. Totuși, utilizatorii sunt interesați de calitatea serviciilor așa că trebuie acordată atenție acestui aspect. Particularitatea rețelelor ad-hoc și alte aspecte legate de rutare și calitatea serviciilor sunt discutate în acest articol.

Comer, *Internetworking with TCP/IP,* Vol.1, ediția a 4-a

Comer a scris cartea decisivă despre suita de protocoale TCP/IP. Capitolele de la 4 la 11 tratează IP și protocoalele legate de el din nivelul rețea. Celelalte capitole se referă în primul rând la nivelele superioare și merită, de asemenea, să fie citite.

Huitema, Routing în the Internet

Dacă vreți să știți tot ce este de știut despre dirijarea în Internet, aceasta este cartea care vă trebuie. Sunt tratați în mare detaliu atât algoritmi care pot fi pronunțați (ca RIP, CIDR și MBONE) cât și algoritmi care nu pot fi pronunțați (ca OSPF, IGRP, EGP și BGP). Se găsesc aici caracteristici noi, cum ar fi trimiterea cu destinație multiplă (multicast), mobil, IP și rezervarea resurselor

Malhotra, *IP Routing*

Lucrarea conține foarte multă informație, constituindu-se într-un îndrumar detaliat asupra rutării IP. Sunt discutate protocoale ca RIP, RIP-2, IGRP, EIGRP, OSPF și BGP-4 .

Metz, „Differentiated Services”

Garanțiile oferite de conceptul de calitate a serviciilor sunt importante pentru multe aplicații multimedia. Serviciile integrate și serviciile diferențiate sunt două abordări posibile pentru obținerea acestor garanții. Ambele sunt discutate aici, cu accent pe serviciile diferențiate.

Metz, „IP Routers: New Tool for Gigabit Networking”

Majoritatea referințelor bibliografice pentru capitolul 5 se referă la algoritmi de rutare. Nu și aceasta, care explică modul de funcționare al ruterelor. Ruterele au evoluat de la stații de lucru pentru uz general până la mașini dedicate pentru rutare. Dacă doriți să aflați mai multe despre rutere, acest articol este un bun punct de pornire.

Nemeth s.a, *UNIX System Administration Handbook*

Pentru o schimbare de ritm, capitolul 13 din această lucrare abordează o viziune mai practică asupra rețelelor decât cele mai multe dintre celelalte referințe bibliografice. În loc să discute concepte abstracte, ea oferă multe sfaturi despre cum se administrează o rețea reală.

Perkins, „Mobile Networking through Mobile IP"

Pe măsură ce dispozitivele mobile de calcul devin din ce în ce mai populare, noțiunea de IP mobil devine din ce în ce mai importantă. Acest îndrumar practic oferă o introducere în acest domeniu discutând și alte subiecte adiacente.

Perlman, *Interconnections: Bridges and Routers,* ediția a 2-a

În capitolele de la 12 pana la 25, Perlman expune multe probleme ce apar în proiectarea algoritmilor de rutare unicast și multicast, atât pentru WAN-uri cât și pentru LAN-uri interconectate, precum și implementările acestor algoritmi în diverse protocoale. Cea mai interesantă parte este însă capitolul 18, în care autoarea analizează, într-o manieră instructivă si amuzantă, anii de experiență în protocoale de rețea.

Puzmanova, *Routing and Switching: Time of Convergence?*

La sfârșitul anilor 1990, unii producători de echipamente de rețea au început să denumească orice drept comutator, în timp ce administratorii de rețele de mari dimensiuni afirmau că au început procesul de trecere de la rutere la comutatoare. Așa cum sugerează și titlul, lucrarea discută despre viitorul ruterelor și comutatoarelor, întrebându-se dacă aceste două echipamente vor converge către un punct comun.

Royer și Toth, „A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks"

Algoritmul de rutare ad-hoc AODV, prezentat in capitolul 5 nu este singurul cunoscut. O diversitate de alți algoritmi, printre care DSDV, CGSR, WRP, DSR, TORA, ABR, DRP și SRP sunt prezentați și comparați în această lucrare. Evident, dacă vă propuneți să inventați un nou algoritm de rutare ad-hoc, primul pas este alegerea unui acronim de trei litere pentru denumire.

Stevens, *TCP/IP Illustrated*, Vol. 1

Capitolele 3-10 furnizează o tratare cuprinzătoare, ilustrată prin exemple, a IP-ului și a protocoalelor legate de el (ARP, RARP și ICMP).

Striegel și Manimaran, „A Survey of QoS Multicasting Issues"

Noțiunile de multicasting și calitate a serviciilor capătă o importanță din ce în ce mai mare pe măsură ce iau amploare servicii ca radioul și televiziunea prin internet. În acest studiu, autorii discută modul în care algoritmii de rutare pot ține cont de acești doi factori.

Yang și Reddy, „A Taxonomy for Congestion Control Algorithms în Packet Switching Networks"

Autorii au proiectat o taxonomie pentru algoritmii de control al congestiei. Principalele categorii sunt buclă deschisă cu controlul sursei, buclă deschisă cu controlul destinației, buclă închisă cu reacție explicită și buclă închisă cu reacție implicită. Ei folosesc această taxonomie pentru a descrie și clasifica 23 de algoritmi existenți.

9.1.6 Nivelul transport

Comer, *Internetworking with TCP/IP*, Vol. 1, ediția a 4-a

Așa cum s-a mai menționat, Comer a scris lucrarea decisivă despre suita de protocoale TCP/IP. Capitolul 12 se referă la UDP, iar capitolul 13 la TCP.

Hall si Cerf, *Internet Core Protocols: The Definitive Guide*

Daca vă place să obțineți informația direct de la sursă, atunci acesta este locul potrivit pentru a afla mai multe despre TCP. În fond, Cerf este unul dintre inventatorii TCP-ului. Capitolul 7 este o prezentare exactă a TCP, în care se arată cum trebuie interpretată informația furnizată de uneltele de analiză a protocoalelor și management al rețelelor. Alte capitole expun aspecte legate de UDP, IGMP, ICMP și ARP.

Kurose si Ross, *Computer Networking: A Top-Down Approach Featuuring the Internet*

Capitolul 3 se referă la nivelul transport și conține o cantitate importantă de informație despre TCP și UDP. Sunt discutate de asemenea protocoalele start-stop și cu revenire cu n pași examinate în capitolul 3.

Mogul, „IP Network Performance”

În ciuda titlului acestui articol, el tratează cel puțin, dacă nu ceva mai mult, performanțele TCP-ului și ale rețelelor în general, apoi performanțele IP-ului în particular. Este plin de îndrumări utile și reguli de aur.

Peterson si Davie, *Computer Networks: A Systems Approach*

Capitolul 5 se referă la UDP, TCP și alte câteva protocoale înrudite. Sunt tratate pe scurt și aspecte legate de performanțele rețelelor.

Stevens, TCP/IP Illustrated, Vol. 1

Capitolele 17-24 furnizează o tratare cuprinzătoare a TCP-ului, ilustrată prin exemple.

9.1.7 Nivelul aplicație

Bergholz, „Extending Your Markup: An XML Tutorial”

O introducere scurtă și directă în XML pentru începători.

Cardellini et. al, *The State-of-the-Art in Locally Distributed Web-Server Systems*

Pe măsură ce serviciul Web crește în popularitate, unele situri Web sunt nevoite să utilizeze un număr mare de servere pentru a putea face față traficului. Partea dificilă în construirea unei ferme de servere o constituie distribuirea încărcării între mașini. Acest îndrumar practic acoperă acest aspect în detaliu.

Berners-Lee et. al, „The World Wide Web”

O perspectivă asupra Web-ului și a direcției sale de evoluție prezentată de persoana care l-a inventat și de câțiva colegi de la CERN. Articolul se concentrează asupra arhitecturii Web, URL-uri, HTTP și HTML, cât și asupra direcțiilor de dezvoltare viitoare.

Choudbury et. al, „Copyright Protection for Electronic Publishing on Computer Networks”

Deși numeroase cărți și articole descriu algoritmi criptografici, puține descriu modul în care aceștia pot fi folosiți pentru a împiedica utilizatorii să redistribuie documentele pe care au dreptul să le decripteze. Acest articol descrie o varietate de mecanisme care poate ajuta protecția drepturilor de autor în era elecrtronică.

Collins, „Carrier Grade Voice over IP”

Dacă ați citit lucrarea scrisă de Varshney et. al și doriti să aflați toate detaliile despre transmisia de voce peste IP utilizând protocolul H.323, acesta este locul potrivit. Deși cartea este lungă și cuprinde multe detalii, este concepută ca un îndrumar practic și nu necesită cunoștințe anterioare de ingineria sistemelor telefonice.

Davison, „A Web Caching Primer”

Pe măsură ce Web-ul crește în dimensiuni, tehnica de ascundere (caching) devine esențială pentru a obține performanțe bune. Lucrarea se constituie într-o scurtă introducere în “Web caching”.

Krishnamurthy și Rexford, *Web Protocols and Practice*

Este dificil să găsești o lucrare mai cuprinzătoare despre toate aspectele Web-ului decât aceasta. Sunt tratate aspecte legate de clienți, servere, proxy-uri și ascundere. Există de asemenea capitole despre măsurători de trafic Web cât și despre cercetările prezente în acest domeniu.

Rabinovich si Spatscheck, *Web Caching and Replication*

Lucrarea este o investiție bună pentru o tratare cuprinzătoare a conceptelor de “Web caching” și replicare. Proxy-uri, cache-uri, încărcare, rețele cu livrare după conținut, selecția serverelor și multe altele sunt prezentate în detaliu.

Shahabi et. al, „Yima: A Second-Generation Continuous Media Server”

Serverele multimedia reprezintă sisteme complexe ce trebuie să rezolve probleme ca planificarea procesoarelor, localizarea fișierelor pe disc, sincronizarea fluxurilor și altele. Cu trecerea timpului, s-au făcut progrese în proiectarea lor. O imagine de ansamblu asupra arhitecturii recente a unui astfel de sistem este prezentată în această lucrare.

Tittel et. al, *Mastering XHTML*

Două carți într-un singur volum, acoperind cel mai nou standard în materie de limbaje de marcaj. Mai întâi este descris limbajul XHTML, cu accent asupra diferențelor față de HTML, apoi este prezentat un ghid cuprinzător al etichetelor, codurilor și caracterelor speciale utilizate în XHTML 1.0.

Varshney et. al, „Voice over IP”

Cum funcționează serviciul de voce peste IP și dacă va înlocui această tehnologie rețeaua comutată de telefonie publică puteți afla citind această lucrare.

9.1.8 Securitatea rețelelor

Anderson, R., „Why Cryptosystems Fail”

După părerea lui Anderson, securitatea sistemelor bancare este firavă, dar nu datorită intrușilor inteligenți care sparg DES-ul pe PC-urile lor. Problemele reale sunt foarte variate, de la angajați necinstiți (un angajat de bancă schimbând adresa de poștă a unui client cu a sa, pentru a intercepta

numărul cărții de credit și numărul de PIN) la erori de programare (acordarea aceluiași cod PIN tuturor clienților). Este deosebit de interesant răspunsul dat de bancă atunci când se confruntă cu o eroare: sistemul nostru este perfect și, ca urmare, toate greșelile trebuie să se datoreze erorilor sau fraudelor clienților.

Anderson, *Security Engineering*

Într-o anumită măsură, această carte este versiunea de 600 de pagini a lucrării „Why Cryptosystems Fail". Are o abordare mai tehnică decât *Secrets and Lies* dar mai puțin tehnică decât *Network Security* (descrisă mai jos). După o introducere a tehnicilor de bază de securitate, sunt dedicate capitole întregi diverselor aplicații, printre care aplicații bancare, comanda și controlul instalațiilor nucleare, imprimarea documentelor secrete, biometrie, securitatea fizică, războiul electronic securitatea telecomunicațiilor, comerț electronic și protecția copyright-ului. A treia parte a lucrării se referă la politici, management și evoluția sistemului.

Artz, „Digital Steganography"

Steganografia își are originile în Grecia antică, unde ceara era folosită pentru a acoperi inscripțiile făcute pe planșe de lemn, asigurându-se astfel secretul mesajelor. În prezent, sunt utilizate alte tehnici, dar scopul rămâne același. Există diverse metode de a ascunde informația în imagini, fluxuri audio și alte purtătoare despre care se vorbește în această lucrare.

Brands, *Rethinking Public Key Infrastructures and Digital Certificates*

Mai mult decât o introducere cuprinzătoare în utilizarea certificatelor digitale, aceasta este și o importantă lucrare de avocatură. Autorul are convingerea că documentele de identitate pe suport de hârtie sunt depășite și ineficiente și că certificatele digitale pot fi utilizate pentru aplicații ca votul electronic, administrarea drepturilor digitale și chiar ca înlocuitor pentru bani gheață. Autorul atrage atenția asupra faptului că, dacă nu se folosește criptarea PKI, Internetul ar putea deveni un instrument de supraveghere pe scară largă.

Kaufman et. al, *Network Security* ediția a 2-a

Această carte demnă de încredere și adesea spirituală este primul loc unde trebuie să căutați informații despre securitatea rețelelor, despre algoritmii si protocoalele utilizate. Se explică pe larg și cu multe exemple algoritmi și protocoale cu chei secrete și publice, criptarea mesajelor, autentificarea, Kerberos, PKI, Ipsec, SSL/TSL și securitatea poștei electronice. Capitolul 26 care tratează subiecte de securitate într-o maniera „folclorică" este o adevarată nestemată. În domeniul securității, diavolul se ascunde în detalii. Oricine își propune să proiecteze un sistem de securitate care va fi utilizat în practică are ce învăța din sfaturile ancorate în realitate prezente în acest capitol.

Pohlmann, *Firewall Systems*

Zidurile de protecție (firewall) reprezintă pentru cele mai multe rețele prima (și ultima) linie defensivă împotriva atacatorilor. Această carte descrie cum funcționează și ce face un zid de protecție, de la cele mai simple ziduri de protecție bazate pe software menite să protejeze un singur PC până la echipamente avansate ce sunt situate între o rețea privată și conexiunea la Internet.

Schneier, *Applied Criptography*, ediția a 2-a

Acest compendiu monumental este cel mai mare coșmar al NSA: o singură carte care descrie fiecare algoritm criptografic cunoscut. Pentru a face situația și mai rea (sau mai bună, depinde de punctul dumneavoastră de vedere) cartea conține cei mai mulți algoritmi ca programe ce pot fi exe-

cutate (în C). Mai mult, sunt furnizate peste 1600 de referințe la literatura criptografică. Deși nu este pentru începători, dacă vreți cu adevărat să păstrați secrete fișierele dumneavoastră, atunci citiți această carte.

Schneier, *Secrets and Lies*

Dacă citiți *Applied Criptography* din scoarță în scoarță, veți afla tot ce este de știut referitor la algoritmi criptografici. Dacă mai citiți apoi și *Secrets and Lies* din scoarță în scoarță (ceea ce nu poate fi făcut într-un timp scurt), veți afla că algoritmii criptografici nu reprezintă totul. Majoritatea breșelor de securitate nu se datorează unor algoritmi greșiti sau cheilor de criptare prea scurte, ci scurgerilor din mediul securizat. Sunt prezentate exemple fără sfârșit despre amenințări, atacuri, defensivă, contraatacuri și multe altele. Pentru o expunere non-tehnică și fascinantă asupra securității computerelor în cel mai larg sens posibil, aceasta este cartea potrivită.

Skoudis, *Counter Hack*

Cel mai bun mod de a opri un hacker este să gândești ca un hacker. Această carte descrie viziunea hacker-ilor asupra rețelelor și argumentează că securitatea ar trebui să fie parte integrantă din proiectarea unei rețele și nu un adaos bazat pe o anumită tehnologie. Lucrarea acoperă toate tipurile uzuale de atacuri, printre care cele de tip „inginerie socială" care profită de faptul că utilizatorii nu sunt întotdeauna familiari cu măsurile de securitate necesare în utilizarea calculatoarelor.

9.2 BIBLIOGRAFIE ÎN ORDINE ALFABETICĂ

ABRAMSON, N.: "Internet Access Using VSATs", *IEEE Commun. Magazine*, vol. 38, pag. 60-68, Iulie 2000.

ABRAMSON, N.: "Development of the ALOHANET", *IEEE Trans. on Information Theory*, vol. IT-31, pag. 119-123, Martie 1985.

ADAMS, M. și DULCHINOS, D.: "OpenCable", *IEEE Commun. Magazine*, vol. 39, pag. 98-105, Iunie 2001.

ALKHATIB, H.S., BAILEY, C., GERLA, M. și McRAE, J.: "Wireless Data Networks: Reaching the Extra Mile", *Computer*, vol. 30, pag. 59-62, Dec. 1997.

ANDERSON, R.J.: "Free Speech Online and Office", *Computer*, vol. 25, pag. 28-30, Iunie 2002.

ANDERSON, R.J.: *Security Engineering*, New York: Wiley, 2001.

ANDERSON, R.J.: "The Eternity Service", *Proc. First Int'l Conf. on Theory and Appl. of Cryptology*, CTU Publishing House, 1996.

ANDERSON, R.J.: "Why Cryptosystems Fail", *Commun. of the ACM*, vol. 37, pag. 32-40, Nov. 1994.

ARTZ, D.: "Digital Steganography", *IEEE Internet Computing*, vol. 5, pag. 75-80, 2001.

AZZAM, A.A. și RANSOM, N.: *Broadband Access Technologies*, New York: McGraw-Hill, 1999.

BAKNE, A. și BADRINATH, B.R.: "I-TCP: Indirect TCP for Mobile Hosts", *Proc. 15th Int'l Conf. on Distr. Computer Systems*, IEEE, pag. 136-143, 1995.

BALAKRISHNAN, H., SESHAN, S. și KATZ, R.H.: "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", *Proc. ACM Mobile Computing and Networking Conf.*, ACM, pag. 2-11, 1995.

BALLARDIE, T., FRANCIS, P. și CROWCROFT, J.: "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.*, ACM, pag. 85-95, 1993.

BARAKAT, C., ALTMAN, E. și DABBOUS, W.: "On TCP Performance in a Heterogeneous Network: A Survey", *IEEE Commun. Magazine*, vol. 38, pag. 40-46, Ian. 2000.

BELLAMY, J.: *Digital Telephon*y, ediția a treia, New York: Wiley, 2000.

BELLMAN, R.E.: *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

BELSNES, D.: "Flow Control in the Packet Switching Networks", *Communications Networks*, Uxbridge, England: Online, pag. 349-361, 1975.

BENNET, C.H. și BRASSARD, G.: "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing*, pag. 175-179, 1984.

BEREZDIVIN, R., BREINIG, R. și TOPP, R.: "Next-Generation Wireless Communication Concepts and Technologies", *IEEE Commun. Magazin*e, vol. 40, pag. 108-116, Martie 2002.

BERGHEL, H.L.: "Cyber Privacy in the New Millennium", *Computer*, vol. 34, pag. 132-134, Ian. 2001.

BERGHOLZ, A.: "Extending Your Markup: An XML Tutorial", *IEEE Internet Computing*, vol. 4, pag. 74-79, Iulie-Aug. 2000.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F. și SECRET, A.: "The World Wide Web", *Commun. of the ACM*, vol. 37, pag. 76-82, Aug. 1994.

BERTSEKAS, D. și GALLAGER, R.: *Data Networks*, ediția a doua, Englewood Cliffs, NJ: Prentice Hall, 1992.

BHAGWAT, P.: "Bluetooth: Technology for Short-Range Wireless Apps", *IEEE Internet Computing*, vol. 5, pag. 96-103, Mai-Iunie 2001.

BHARGHAVAN, V., DEMERS, A., SHENKER, S. și ZHANG, L.: "MACAW: A Media Access Protocol for Wireless LANs", *Proc. SIGCOMM '94 Conf.*, ACM, pag. 212-225, 1994.

BHATTI, S.N. și CROWCROFT, J.: "QoS Sensitive Flows: Issues in IP Packet Handling, "*IEEE Internet Computing*, vol. 4, pag. 48-57, Iulie-Aug. 2000.

BI, Q., ZYSMAN, G.I. și MENKES, H.: "Wireless Mobile Communications at the Start of the 21st Century", *IEEE Commun. Magazine*, vol. 39, pag. 110-116, Jan, 2001.

BIHAM, E. și SHAMIR, A.: "Differential Cryptanalysis of the Data Encryption Standard, "*Proc. 17th Ann. Int'l Cryptology Conf.*, Berlin: Springer-Verlag LNCS 1294, pag. 513-525, 1997.

BIRD, R., GOPAL, I., HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R şi YUNG, M.: "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.*, vol. 11, pag. 679-693, Iunie 1993.

BIRRELL, A.D. şi NELSON, B.J.: "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems*, vol. 2, pag. 39-59, Feb. 1984.

BIRYUKOV, A., SHAMIR, A. şi WAGNER, D.: "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption*, Berlin: Springer-Verlag LNCS 1978, p. 1, 2000.

BISDIKIAN, C.: "An Overview of the Bluetooth Wireless Technology", *IEEE Commun. Magazin*e, vol. 39, pag. 86-94, Dec. 2001.

BLAZE, M.: "Protocol Failure in the Escrowed Encryption Standard", *Proc. Second ACM Conf. on Computer and Commun. Security*, ACM, pag. 59-67, 1994.

BLAZE, M. şi BELLOVIN, S.: "Tapping on My Network Door", *Commun. of the ACM*, vol. 43, pag. 136, Oct. 2000.

BOGINENI, K., SIVALINGAM, K.M. şi DOWD, P.W.: "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks", *IEEE Journal on Selected Areas in Commun.*, vol. 11, pag. 590-604, Mai 1993.

BOLCSKEI, H., PAULRAJ, A.J., HARI, K.V.S. şi NABAR, R.U.: "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Commun. Magazine*, vol. 39, pag. 100-108, Ian. 2001.

BORISOV, N., GOLDBERG, I. şi WAGNER, D.: "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, pag. 180-188, 2001.

BRANDS, S.: *Rethinking Public Key Infrastructures and Digital Certificates*, Cambridge, MA: M.I.T. Press, 2000.

BRAY, J. şi STURMAN, C.F.: *Bluetooth 1.1: Connect without Cables*, ediția a doua, Upper Saddle River, NJ: Prentice Hall, 2002.

BREYER, R. şi RILEY, S.: *Switched, Fast, and Gigabit Ethernet*, Indianapolis, IN: New Riders, 1999.

BROWN, S.: *Implementing Virtual Private Networks*, New York: McGraw-Hill, 1999.

BROWN, L., KWAN, M., PIEPRZYK, J. şi SEBERRY, J.: "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI", *ASIACRYPT '91 Abstracts*, pag. 25-30, 1991.

BURNETT, S. şi PAINE, S.: *RSA Security's Official Guide to Cryptography*, Berkeley, CA: Osborne/McGraw-Hill, 2001.

CAPETANAKIS, J.I.: "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory*, vol. IT-25, pag. 505-515, Sept. 1979.

CARDELLINI, V., CASALICCHIO, E., COLAJANNI, M. şi YU, P.S.: *"The State-of-the-Art in Locally Distributed Web-Server Systems", ACM Computing Surveys, vol. 34, pag. 263-311, Iunie 2002.*

CARLSON, J.: *PPP Design, Implementation and Debugging*, ediția a doua, Boston: Addison-Wesley, 2001.

CERF, V. și KAHN, R.: "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.*, vol. COM-22, pag. 637-648, Mai 1974.

CHAKRABARTI, S.: "QoS Issues in Ad Hoc Wireless Networks", *IEEE Commun. Magazine*, vol. 39, pag. 142-148, Feb. 2001.

CHASE, J.S., GALLATIN, A.J. și YOCUM, K.G.: "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine*, vol. 39, pag. 68-75, Aprilie 2001.

CHEN, B., JAMIESON, K., BALAKRISHNAN, H. și MORRIS, R.: "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Net-works, "*ACM Wireless Networks*, vol. 8, Sept. 2002.

CHEN, K.-C.: "Medium Access Control of Wireless LANs for Mobile Computing", *IEEE Network Magazine*, vol. 8, pag. 50-63, Sept.-Oct. 1994.

CHOUDBURY, A.K., MAXEMCHUK, N.F., PAUL, S. și SCHULZRINNE, H.G.: "Copy-right Protection for Electronic Publishing on Computer Networks", *IEEE Network Magazine*, vol. 9, pag. 12-20, Mai-Iunie, 1995.

CHU, Y., RAO, S.G. și ZHANG, H.: "A Case for End System Multicast", *Proc. Int'l Conf. on Measurements and Modeling of Computer Syst.*, ACM, pag. 1-12, 2000.

CLARK, D.D.: "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.*, ACM, pag. 106-114, 1988.

CLARK, D.D.: "Window and Acknowledgement Strategy in TCP", RFC 813, Iulie 1982.

CLARK, D.D., DAVIE, B.S., FARBER, D.J., GOPAL, I.S., KADABA, B.K., SINCOSKIE, W.D., SMITH, J.M. și TENNENHOUSE, D.L.: "The Aurora Gigabit Testbed", *Com-puter Networks and ISDN Systems*, vol. 25, pag. 599-621, Ian. 1993.

CLARK, D.D., JACOBSON, V., ROMKEY, J. și SALWEN, H.: "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine*, vol. 27, pag. 23-29, Iunie 1989.

CLARK, D.D., LAMBERT, M. și ZHANG, L.: "NETBLT: A High Throughput Transport Protocol", *Proc. SIGCOMM '87 Conf.*, ACM, pag. 353-359, 1987.

CLARKE, A.C.: "Extra-Terrestrial Relays", *Wireless Worl*d, 1945.

CLARKE, I., MILLER, S.G., HONG, T.W., SANDBERG, O. și WILEY, B.: "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, vol. 6, pag. 40-49, Ian.-Feb. 2002.

COLLINS, D.: *Carrier Grade Voice over IP*, New York: McGraw-Hill, 2001.

COLLINS, D. și SMITH, C.: *3G Wireless Network*s, New York: McGraw-Hill, 2001.

COMER, D.E.: *The Internet Book*, Englewood Cliffs, NJ: Prentice Hall, 1995.

COMER, D.E.: *Internetworking with TCP/IP*, vol. 1, ediția a patra, Englewood Cliffs, NJ: Pren-tice Hall, 2000.

COSTA, L.H.M.K., FDIDA, S. și DUARTE, O.C.M.B.: "Hop by Hop Multicast Routing Protocol", *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Proto-cols for Computer Commun.*, ACM, pag. 249-259, 2001.

CRAVER, S.A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D. și FELTEN, E.W.: "Reading Between the Lines: Lessons from the SDMI Challenge", *Proc. 10th USENIX Security Symp.*, USENIX, 2001.

CRESPO, P.M., HONIG, M.L. și SALEHI, J.A.: "Spread-Time Code-Division Multiple Access", *IEEE Trans. on Commun.*, vol. 43, pag. 2139-2148, Iunie 1995.

CROW, B.P., WIDJAJA, I, KIM, J.G. și SAKAI, P.T.: "IEEE 802.11 Wireless Local Area Networks", *IEEE Commun. Magazine*, vol. 35, pag. 116-126, Sept. 1997

CROWCROFT, J., WANG, Z., SMITH, A. și ADAMS, J.: "A Rough Comparison of the IETF and ATM Service Models", *IEEE Network Magazine*, vol. 9, pag. 12-16, Nov.-Dec. 1995.

DABEK, F., BRUNSKILL, E., KAASHOEK, M.F., KARGER, D., MORRIS, R., STOICA, R. și BALAKRISH-NAN, H.: "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service", *Proc. 8th Workshop on Hot Topics in Operating Systems*, IEEE, pag. 71-76, 2001a.

DABEK, F., KAASHOEK, M.F., KARGER, D., MORRIS, R. și STOICA, I.: "Wide-Area Cooperative Stor-age with CFS", *Proc. 18th Symp. on Operating Systems Prin.*, ACM, pag. 202-15 , 2001b.

DAEMEN, J. și RIJMEN, V.: *The Design of Rijndael*, Berlin: Springer-Verlag, 2002.

DANTHINE, A.A.S.: "Protocol Representation with Finite-State Models", *IEEE Trans. on Commun.*, vol. COM-28, pag. 632-643, Aprilie 1980.

DAVIDSON, J. și PETERS, J.: *Voice over IP Fundamentals*, Indianapolis, IN: Cisco Press, 2000.

DAVIE, B. și REKHTER, Y.: *MPLS Technology and Applications*, San Francisco: Mor-gan Kauf-mann, 2000.

DAVIS, P.T. și McGUFFIN, C.R.: *Wireless Local Area Networks*, New York: McGraw-Hill, 1995.

DAVISON, B.D.: "A Web Caching Primer", *IEEE Internet Computing*, vol. 5, pag. 38-45, Iulie-Aug. 2001.

DAY, J.D.: " The (Un)Revised OSI Reference Model", *Computer Commun. Rev.*, vol. 25, pag. 39-55, Oct. 1995.

DAY, J.D. și ZIMMERMANN, H.: "The OSI Reference Model", *Proc. of the IEEE*, vol. 71, pag. 1334-1340, Dec. 1983.

DE VRIENDT, J., LAINE, P., LEROUGE, C și XU, X.: "Mobile Network Evolution: A Revolution on the Move", *IEEE Commun. Magazine*, vol. 40, pag. 104-111, Aprilie 2002.

DEERING, S.E.: "SIP: Simple Internet Protocol", *IEEE Network Magazine*, vol. 7, pag. 16-28, Mai-Iunie 1993.

DEMERS, A., KESHAV, S. și SHENKER, S.: "Analysis and Simulation of a Fair Queue-ing Algorithm", *Internetwork: Research and Experience*, vol. 1, pag. 3-26, Sept. 1990.

DENNING, D.E. și SACCO, G.M.: "Timestamps in Key Distribution Protocols", *Com-mun. of the ACM*, vol. 24, pag. 533-536, Aug. 1981.

DIFFIE, W. și HELLMAN, M.E.: "Exhaustive Cryptanalysis of the NBS Data Encryp-tion Standard", *Computer*, vol. 10, pag. 74-84, Iunie 1977.

DIFFIE, W. și HELLMAN, M.E.: "New Directions in Cryptography", *IEEE Trans. on Information Theory*, vol. IT-22, pag. 644-654, Nov. 1976.

DIJKSTRA, E.W.: "A Note on Two Problems in Connexion with Graphs", *Numer. Math.*, vol. 1, pag. 269-271, Oct. 1959.

DOBROWSKI, G. și GRISE, D.: *ATM and SONET Basics*, Fuquay-Varina, NC: APDG Telecom Books, 2001.

DONALDSON, G. și JONES, D.: "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine*, vol. 39, pag. 122-126, Iunie 2001.

DORFMAN, R.: "Detection of Defective Members of a Large Population", *Annals Math. Statistics*, vol. 14, pag. 436-440, 1943.

DOUFEXI, A., ARMOUR, S., BUTLER, M., NIX, A., BULL, D., MCGEEHAN, J. și KARLSSON, P.: "A Comparison of the HIPERLAN/2 and IEEE 802.11A Wireless LAN Standards", *IEEE Commun. Magazine*, vol. 40, pag. 172-180, Mai 2002.

DURAND, A.: "Deploying IPv6", *IEEE Internet Computing*, vol. 5, pag. 79-81, Ian.-Feb. 2001.

DUTCHER, B.: *The NAT Handbook*, New York: Wiley, 2001.

DUTTA-ROY, A.: "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine*, vol. 39, pag. 81-88, Iunie 2001.

EASTTOM, C.: *Learn JavaScript*, Ashburton, U.K.: Wordware Publishing, 2001.

EL GAMAL, T.: "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory*, vol. IT-31, pag. 469-472, Iulie 1985.

ELHANANY, I., KAHANE, M. și SADOT, D.: "Packet Scheduling in Next-Generation Multiterabit Networks", *Computer*, vol. 34, pag. 104-106, Aprilie 2001.

ELMIRGHANI, J.M.H. și MOUFTAH, H.T.: "Technologies and Architectures for Scal-able Dynamic Dense WDM Networks", *IEEE Commun. Magazine*, vol. 38, pag. 58-66, Feb. 2000.

FARSEROTU, J. și PRASAD, R.: "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends", *IEEE Commun. Magazine*, vol. 38, pag. 128-133, Iunie 2000.

FIORINI, D., CHIANI, M., TRALLI, V. și SALATI., C.: "Can we Trust HDLC?", *Com-puter Commun. Rev*., vol. 24, pag. 61-80, Oct. 1994.

FLOYD, S. și JACOBSON, V.: "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking*, vol. 1, pag. 397-413, Aug. 1993.

FLUHRER, S., MANTIN, I. și SHAMIR, A.: "Weakness in the Key Scheduling Algo-rithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography*, 2001.

FORD, L.R., Jr. și FULKERSON, D.R.: *Flows in Networks*, Princeton, NJ: Princeton University Press, 1962.

FORD, W. și BAUM, M.S.: *Secure Electronic Commerce*, Upper Saddle River, NJ: Prentice Hall, 2000.

FORMAN, G.H. și ZAHORJAN, J.: "The Challenges of Mobile Computing", *Computer*, vol. 27, pag. 38-47, Aprilie 1994.

FRANCIS, P.: "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine*, vol. 7, pag. 30-37, Mai-Iunie 1993.

FRASER, A.G.: "Towards a Universal Data Transport System", in *Advances in Local Area Networks*, Kummerle, K., Tobagi, F. și Limb, J.O. (Eds.), New York: IEEE Press, 1987.

FRENGLE, N.: *I-Mode: A Primer*, New York: Hungry Minds, 2002.

GADECKI, C. și HECKERT, C.: *ATM for Dummies*, New York: Hungry Minds, 1997.

GARBER, L.: "Will 3G Really Be the Next Big Wireless Technology?", *Computer*, vol. 35, pag. 26-32, Ian. 2002.

GARFINKEL, S., with SPAFFORD, G.: *Web Security, Privacy, and Commerce*, Sebastopol, CA: O'Reilly, 2002.

GEIER, J.: *Wireless LAN*s, ediția a doua, Indianapolis, IN: Sams, 2002.

GEVROS, P., CROWCROFT, J., KIRSTEIN, P. și BHATTI, S.: "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network Magazine*, vol. 15, pag. 16-25, Mai-Iunie 2001.

GHANI, N. și DIXIT, S.: "TCP/IP Enhancements for Satellite Networks", *IEEE Commun. Magazine*, vol. 37, pag. 64-72, 1999.

GINSBURG, D.: *ATM: Solutions for Enterprise Networking*, Boston: Addison-Wesley, 1996.

GOODMAN, D.J.: "The Wireless Internet: Promises and Challenges", *Computer*, vol. 33, pag. 36-41, Iulie 2000.

GORALSKI, W.J.: *Optical Networking and WDM*, New York: McGraw-Hill, 2001.

GORALSKI, W.J.: *SONET*, ediția a doua, New York: McGraw-Hill, 2000.

GORALSKI, W.J.: *Introduction to ATM Networkin*g, New York: McGraw-Hill, 1995.

GOSSAIN, H., DE MORAIS CORDEIRO și AGRAWAL, D.P.: "Multicast: Wired to Wireless", *IEEE Commun. Mag.*, vol. 40, pag. 116-123, Iunie 2002.

GRAVANO, S.: *Introduction to Error Control Code*s, Oxford, U.K.: Oxford University Press, 2001.

GUO, Y. și CHASKAR, H.: "Class-Based Quality of Service over Air Interfaces in 4G Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pag. 132-137, Martie 2002.

HAARTSEN, J.: "The Bluetooth Radio System", *IEEE Personal Commun. Magazine*, vol. 7, pag. 28-36, Feb. 2000.

HAC, A.: "Wireless and Cellular Architecture and Services", *IEEE Commun. Magazine*, vol. 33, pag. 98-104, Nov. 1995.

HAC, A. și GUO, L.: "A Scalable Mobile Host Protocol for the Internet", *Int'l J. of Network Mgmt*, vol. 10, pag. 115-134, Mai-Iunie, 2000.

HALL, E. și CERF, V.: *Internet Core Protocols: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

HAMMING, R.W.: "Error Detecting and Error Correcting Codes", *Bell System Tech. J.*, vol. 29, pag. 147-160, Aprilie 1950.

HANEGAN, K.: *Custom CGI Scripting with Perl*, New York: Wiley, 2001.

HARRIS, A.: *JavaScript Programming for the Absolute Beginner*, Premier Press, 2001.

HARTE, L., KELLOGG, S., DREHER, R. și SCHAFFNIT, T.: *The Comprehensive Guide to Wireless Technology*, Fuquay-Varina, NC: APDG Publishing, 2000.

HARTE, L., LEVINE, R. și KIKTA, R.: *3G Wireless Demystified*, New York: McGraw-Hill, 2002.

HAWLEY, G.T.: "Historical Perspectives on the U.S. Telephone System", *IEEE Commun. Magazine*, vol. 29, pag. 24-28, Martie 1991.

HECHT, J.: "Understanding Fiber Optics", Upper Saddle River, NJ: Prentice Hall, 2001.

HEEGARD, C., COFFEY, J.T., GUMMADI, S., MURPHY, P.A., PROVENCIO, R., ROSSIN, E.J., SCHRUM, S. și SHOEMAKER, M.B.: "High-Performance Wireless Ethernet", *IEEE Commun. Magazine*, vol. 39, pag. 64-73, Nov. 2001.

HELD, G.: *The Complete Modem Reference*, ediția a doua, New York: Wiley, 1994.

HELLMAN, M.E.: "A Cryptanalytic Time-Memory Tradeoff", *IEEE Trans. on Information Theor*y, vol. IT-26, pag. 401-406, Iulie 1980.

HILLS, A.: "Large-Scale Wireless LAN Design", *IEEE Commun. Magazin*e, vol. 39, pag. 98-104, Nov. 2001.

HOLZMANN, G.J.: *Design and Validation of Computer Protocol*s, Englewood Cliffs, NJ: Prentice Hall, 1991.

HU, Y. și LI, V.O.K.: "Satellite-Based Internet Access", *IEEE Commun. Magazin*e, vol. 39, pag. 155-162, Martie 2001.

HU, Y.-C. și JOHNSON, D.B.: "Implicit Source Routes for On-Demand Ad Hoc Network Routing", *Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking & Computing*, ACM, pag. 1-10, 2001.

HUANG, V. și ZHUANG, W.: "QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications", *IEEE Commun. Magazine*, vol. 40, pag. 118-125, Martie 2002.

HUBER, J.F., WEILER, D. și BRAND, H.: "UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization", *IEEE Commun. Magazine*, vol. 38, pag. 129-136, Sept. 2000. nr u 0

HUI, J.: "A Broadband Packet Switch for Multi-rate Services", *Proc. Int'l Conf. on Commun*., IEEE, pag. 782-788, 1987.

HUITEMA, C.: *Routing in the Internet*, Englewood Cliffs, NJ: Prentice Hall, 1995.

HULL, S.: *Content Delivery Networks*, Berkeley, CA: Osborne/McGraw-Hill, 2002.

HUMBLET, P.A., RAMASWAMI, R. și SIVARAJAN, K.N.: "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks", *Proc. SIGCOMM '92 Conf*., ACM, pag. 2-13, 1992.

HUNTER, D.K. și ANDONOVIC, I.: "Approaches to Optical Internet Packet Switching, " *IEEE Commun. Magazine*, vol. 38, pag. 116-122, Sept. 2000.

HUSTON, G.: "TCP in a Wireless World", *IEEE Internet Computing*, vol. 5, pag. 82-84, Martie-Aprilie, 2001.

IBE, O.C.: *Essentials of ATM Networks and Services*, Boston: Addison-Wesley, 1997.

IRMER, T.: "Shaping Future Telecommunications: The Challenge of Global Standardization, " *IEEE Commun. Magazine*, vol. 32, pag. 20-28, Ian. 1994.

IZZO, P.: *Gigabit Networks*, New York: Wiley, 2000.

JACOBSON, V.: "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf*., ACM, pag. 314-329, 1988.

JAIN, R.: "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey", *Computer Networks and ISDN Systems*, vol. 27, Nov. 1995.

JAIN, R.: *FDDI Handbook—High-Speed Networking Using Fiber and Other Media*, Boston: Addison-Wesley, 1994.

JAIN, R.: "Congestion Control in Computer Networks: Issues and Trends", *IEEE Network Magazine*, vol. 4, pag. 24-30, Mai-Iunie 1990.

JAKOBSSON, M. și WETZEL, S.: "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 2001*, Berlin: Springer-Verlag LNCS 2020, pag. 176-191, 2001.

JOEL, A.: "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine*, 50th Anniversary Issue, pag. 6-14 and 162-167, Mai 2002.

JOHANSSON, P., KAZANTZIDIS, M., KAPOOR, R. și GERLA, M.: "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network Magazine*, vol. 15, pag. 28-37, Sept.-Oct 2001.

JOHNSON, D.B.: "Scalable Support for Transparent Mobile Host Internetworking", *Wireless Networks*, vol. 1, pag. 311-321, Oct. 1995.

JOHNSON, H.W.: *Fast Ethernet—Dawn of a New Network*, Englewood Cliffs, NJ: Prentice Hall, 1996.

JOHNSON, N.F. și JAJODA, S.: "Exploring Steganography: Seeing the Unseen", *Computer*, vol. 31, pag. 26-34, Feb. 1998.

KAHN, D.: "Cryptology Goes Public", *IEEE Commun. Magazine*, vol. 18, pag. 19-28, Martie 1980.

KAHN, D.: *The Codebreakers*, ediția a doua, New York: Macmillan, 1995.

KAMOUN, F. și KLEINROCK, L.: "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks*, vol. 3, pag. 337-353, Nov. 1979.

KAPP, S.: "802.11: Leaving the Wire Behind", *IEEE Internet Computing*, vol. 6, pag. 82- 85, Ian.-Feb. 2002.

KARN, P.: "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, pag. 134-140, 1990.

KARTALOPOULOS, S.: *Introduction to DWDM Technology: Data in a Rainbow*, New York, NY: IEEE Communications Society, 1999.

KASERA, S.K., HJALMTYSSON, G., TOWLSEY, D.F. și KUROSE, J.F.: "Scalable Reliable Multicast Using Multiple Multicast Channels", *IEEE/ACM Trans. on Networking*, vol. 8, pag. 294-310, 2000.

KATZ, D. și FORD, P.S.: "TUBA: Replacing IP with CLNP", *IEEE Network Magazine*, vol. 7, pag. 38-47, Mai-Iunie 1993.

KATZENBEISSER, S. și PETITCOLAS, F.A.P.: *Information Hiding Techniques for Steganography and Digital Watermarking*, London, Artech House, 2000.

KAUFMAN, C., PERLMAN, R. și SPECINER, M.: *Network Security*, ediția a doua, Englewood Cliffs, NJ: Prentice Hall, 2002.

KELLERER, W., VOGEL, H.-J. și STEINBERG, K.-E.: "A Communication Gateway for Infrastructure-Independent 4G Wireless Access", *IEEE Commun. Magazine*, vol. 40, pag. 126-131, Martie 2002.

KERCKHOFF, A.: "La Cryptographie Militaire", *J. des Sciences Militaires*, vol. 9, pag. 5- 38, Ian. 1883 și pag. 161-191, Feb. 1883.

KIM, J.B., SUDA, T. și YOSHIMURA, M.: "International Standardization of B-ISDN", *Computer Networks and ISDN Systems*, vol. 27, pag. 5-27, Oct. 1994.

KIPNIS, J.: "Beating the System: Abuses of the Standards Adoptions Process", *IEEE Commun. Magazine*, vol. 38, pag. 102-105, Iulie 2000.

KLEINROCK, L.: "On Some Principles of Nomadic Computing and Multi-Access Communications, "*IEEE Commun. Magazine*, vol. 38, pag. 46-50, Iulie 2000.

KLEINROCK, L. și TOBAGI, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf*., pag. 187-201, 1975.

KRISHNAMURTHY, B. și REXFORD, J.: *Web Protocols and Practice*, Boston: Addison-Wesley, 2001.

KUMAR, V., KORPI, M. și SENGODAN, S.: *IP Telephony with H.323*, New York: Wiley, 2001.

KUROSE, J.F. și ROSS, K.W.: *Computer Networking: A Top-Down Approach Featuring the Internet*, Boston: Addison-Wesley, 2001.

KWOK, T.: "A Vision for Residential Broadband Service: ATM to the Home", *IEEE Network Magazine*, vol. 9, pag. 14-28, Sept.-Oct. 1995.

KYAS, O. și CRAWFORD, G.: *ATM Networks*, Upper Saddle River, NJ: Prentice Hall, 2002.

LAM, C.K.M. și TAN, B.C.Y.: "The Internet Is Changing the Music Industry", *Commun. of the ACM*, vol. 44, pag. 62-66, Aug. 2001.

LANSFORD, J., STEPHENS, A și NEVO, R.: "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence", *IEEE Network Magazine*, vol. 15, pag. 20-27, Sept.-Oct 2001.

LASH, D.A.: *The Web Wizard's Guide to Perl and CGI*, Boston: Addison-Wesley, 2002.

LAUBACH, M.E., FARBER, D.J. și DUKES, S.D.: *Delivering Internet Connections over Cable*, New York: Wiley, 2001.

LEE, J.S. și MILLER, L.E.: *CDMA Systems Engineering Handbook*, London: Artech House, 1998.

LEEPER, D.G.: "A Long-Term View of Short-Range Wireless", *Computer*, vol. 34, pag. 39-44, Iunie 2001.

LEINER, B.M., COLE, R., POSTEL, J. și MILLS, D.: "The DARPA Internet Protocol Suite", *IEEE Commun. Magazine*, vol. 23, pag. 29-34, Martie 1985.

LEVINE, D.A. și AKYILDIZ, I.A.: "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology", *IEEE/ACM Trans. on Networking*, vol. 3, pag. 158-168, Aprilie 1995.

LEVY, S.: "Crypto Rebels", *Wired*, pag. 54-61, Mai-Iunie 1993.

LI, J., BLAKE, C., DE COUTO, D.S.J., LEE, H.I. și MORRIS, R.: "Capacity of Ad Hoc Wireless Networks", *Proc. 7th Int'l Conf. on Mobile Computing and Networking*, ACM, pag. 61-69, 2001.

LIN, F., CHU, P. și LIU, M.: "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies", *Proc. SIGCOMM '87 Conf*., ACM, pag. 126-135, 1987.

LIN, Y.-D., HSU, N.-B. şi HWANG, R.-H.: "QoS Routing Granularity in MPLS Networks", *IEEE Commun. Magazine*, vol. 40, pag. 58-65, Iunie 2002.

LISTANI, M., ERAMO, V. şi SABELLA, R.: "Architectural and Technological Issues for Future Optical Internet Networks", *IEEE Commun. Magazine*, vol. 38, pag. 82-92, Sept. 2000.

LIU, C.L. şi LAYLAND, J.W.: "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM*, vol. 20, pag. 46-61, Ian. 1973.

LIVINGSTON, D.: *Essential XML for Web Professionals*, Upper Saddle River, NJ: Prentice Hall, 2002.

LOSHIN, P.: *IPv6 Clearly Explained*, San Francisco: Morgan Kaufmann, 1999.

LOUIS, P.J.: *Broadband Crash Course*, New York: McGraw-Hill, 2002.

LU, W.: *Broadband Wireless Mobile: 3G and Beyond*, New York: Wiley, 2002.

MACEDONIA, M.R.: "Distributed File Sharing", *Computer*, vol. 33, pag. 99-101, 2000.

MADRUGA, E.L. şi GARCIA-LUNA-ACEVES, J.J.: "Scalable Multicasting: the Core-Assisted Mesh Protocol", *Mobile Networks and Applications*, vol. 6, pag. 151-165, Aprilie 2001.

MALHOTRA, R.: *IP Routin*g, Sebastopol, CA: O'Reilly, 2002.

MATSUI, M.: "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology— Eurocrypt '93 Proceedings*, Berlin: Springer-Verlag LNCS 765, pag. 386-397, 1994.

MAUFER, T.A.: *IP Fundamentals*, Upper Saddle River, NJ: Prentice Hall, 1999.

MAZIERES, D. şi KAASHOEK, M.F.: "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security*, ACM, pag. 27-36, 1998.

MAZIERES, D., KAMINSKY, M., KAASHOEK, M.F. şi WITCHEL, E.: "Separating Key Management from File System Security", *Proc. 17th Symp. on Operating Systems Prin*., ACM, pag. 124-139, Dec. 1999.

McFEDRIES, P: *Using JavaScrip*t, Indianapolis, IN: Que, 2001.

McKENNEY, P.E. şi DOVE, K.F.: "Efficient Demultiplexing of Incoming TCP Packets, " *Proc. SIGCOMM '92 Conf*., ACM, pag. 269-279, 1992.

MELTZER, K. şi MICHALSKI, B.: *Writing CGI Applications with Perl*, Boston: Addison-Wesley, 2001.

MENEZES, A.J. şi VANSTONE, S.A.: "Elliptic Curve Cryptosystems and Their Implementation, " *Journal of Cryptology*, vol. 6, pag. 209-224, 1993.

MERKLE, R.C.: "Fast Software Encryption Functions", *Advances in Cryptology— CRYPTO '90 Proceedings*, Berlin: Springer-Verlag LNCS 473, pag. 476-501, 1991.

MERKLE, R.C. şi HELLMAN, M.: "On the Security of Multiple Encryption", *Commun. of the AC*M, vol. 24, pag. 465-467, Iulie 1981.

MERKLE, R.C. și HELLMAN, M.: "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory*, vol. IT-24, pag. 525-530, Sept. 1978.

METCALFE, R.M.: "On Mobile Computing", *Byte*, vol. 20, p. 110, Sept. 1995.

METCALFE, R.M.: "Computer/Network Interface Design: Lessons from Arpanet and Ethernet, " *IEEE Journal on Selected Areas in Commun*., vol. 11, pag. 173-179, Feb. 1993.

METCALFE, R.M. și BOGGS, D.R.: "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, pag. 395-404, Iulie 1976.

METZ, C: "Interconnecting ISP Networks", *IEEE Internet Computing*, vol. 5, pag. 74-80, Martie-Aprilie 2001.

METZ, C.: "Differentiated Services", *IEEE Multimedia Magazine*, vol. 7, pag. 84-90, Iulie-Sept. 2000.

METZ, C.: "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing*, vol. 2, pag. 14-18, Nov.-Dec. 1998.

MILLER, B.A. și BISDIKIAN, C.,: *Bluetooth Revealed*, Upper Saddle River, NJ: Prentice Hall, 2001.

MILLER, P. și CUMMINS, M.: *LAN Technologies Explained*, Woburn, MA: Butterworth-Heinemann, 2000.

MINOLI, D.: *Video Dialtone Technology*, New York: McGraw-Hill, 1995.

MINOLI, D. și VITELLA, M.: *ATM & Cell Relay for Corporate Environments*, New York: McGraw-Hill, 1994.

MISHRA, P.P. și KANAKIA, H.: "A Hop by Hop Rate-Based Congestion Control Scheme", *Proc. SIGCOMM '92 Conf*., ACM, pag. 112-123, 1992.

MISRA, A., DAS, S., DUTTA, A., McAULEY, A. și DAS, S.: "IDMP-Based Fast Handoffs and Paging in IP-Based 4G Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pag. 138-145, Martie 2002.

MOGUL, J.C.: "IP Network Performance", in *Internet System Handbook*, Lynch, D.C. and Rose, M.T. (eds.), Boston: Addison-Wesley, pag. 575-675, 1993.

MOK, A.K. și WARD, S.A.: "Distributed Broadcast Channel Access", *Computer Networks*, vol. 3, pag. 327-335, Nov. 1979.

MOY, J.: "Multicast Routing Extensions", *Commun. of the ACM*, vol. 37, pag. 61-66, Aug. 1994.

MULLINS, J.: "Making Unbreakable Code", *IEEE Spectrum*, pag. 40-45, Mai 2002.

NAGLE, J.: "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun*., vol. COM-35, pag. 435-438, Aprilie 1987.

NAGLE, J.: "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev*., vol. 14, pag. 11-17, Oct. 1984.

NARAYANASWAMI, C., KAMIJOH, N., RAGHUNATH, M., INOUE, T., CIPOLLA, T., SANFORD, J., SCHLIG, E., VENTKITESWARAN, S., GUNIGUNTALA, D., KUL-KARNI, V. și YAMAZAKI, K.: "IBM's Linux Watch: The Challenge of Miniaturization", *Computer*, vol. 35, pag. 33-41, Ian. 2002.

NAUGHTON, J.: "A Brief History of the Future", Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R.M. și SCHROEDER, M.D.: "Authentication Revisited", *Operating Systems Rev*., vol. 21, p. 7, Ian. 1987.

NEEDHAM, R.M. și SCHROEDER, M.D.: "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM*, vol. 21, pag. 993-999, Dec. 1978.

NELAKUDITI, S. și ZHANG, Z.-L.: "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine* vol. 40, pag. 66-71, Iunie 2002.

NEMETH, E., SNYDER, G., SEEBASS, S. și HEIN, T.R.: *UNIX System Administration Handbook*, ediția a treia, Englewood Cliffs, NJ: Prentice Hall, 2000.

NICHOLS, R.K. și LEKKAS, P.C.: *Wireless Security*, New York: McGraw-Hill, 2002.

NIST: "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

O'HARA, B. și PETRICK, A.: *802.11 Handbook: A Designer's Companion*, New York: IEEE Press, 1999.

OTWAY, D. și REES, O.: "Efficient and Timely Mutual Authentication", *Operating Systems Rev*., pag. 8-10, Ian. 1987.

OVADIA, S.: *Broadband Cable TV Access Networks: from Technologies to Applications*, Upper Saddle River, NJ: Prentice Hall, 2001.

PALAIS, J.C.: *Fiber Optic Commun*., ediția a treia, Englewood Cliffs, NJ: Prentice Hall, 1992.

PAN, D.: "A Tutorial on MPEG/Audio Compression", *IEEE Multimedia Magazine*, vol. 2, pag.60-74, Summer 1995.

PANDYA, R.: "Emerging Mobile and Personal Communication Systems", *IEEE Commun. Magazine*, vol. 33, pag. 44-52, Iunie 1995.

PARAMESWARAN, M., SUSARLA, A. și WHINSTON, A.B.: "P2P Networking: An Information-Sharing Alternative", *Computer*, vol. 34, pag. 31-38, Iulie 2001.

PARK, J.S. și SANDHU, R.: "Secure Cookies on the Web", *IEEE Internet Computing*, vol. 4, pag. 36-44, Iulie-Aug. 2000.

PARTRIDGE, C., HUGHES, J. și STONE, J.: "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf*., ACM, pag. 68-76, 1995.

PAXSON, V.: "Growth Trends in Wide-Area TCP Connections", *IEEE Network Magazine*, vol. 8, pag. 8-17, Iulie-Aug. 1994.

PAXSON, V. și FLOYD, S.: "Wide-Area Traffic: The Failure of Poisson Modeling", *Proc. SIGCOMM '94 Conf.*, ACM, pag. 257-268, 1995.

PEPELNJAK, I. și GUICHARD, J.: *MPLS and VPN Architectures*, Indianapolis, IN: Cisco Press, 2001.

PERKINS, C.E.: *RTP: Audio and Video for the Internet*, Boston: Addison-Wesley, 2002.

PERKINS, C.E. (ed.): *Ad Hoc Networkin*g, Boston: Addison-Wesley, 2001.

PERKINS, C.E.: *Mobile IP Design Principles and Practices*, Upper Saddle River, NJ: Prentice Hall, 1998a.

PERKINS, C.E.: "Mobile Networking in the Internet", *Mobile Networks and Applications*, vol. 3, pag. 319-334, 1998b.

PERKINS, C.E.: "Mobile Networking through Mobile IP", *IEEE Internet Computing*, vol. 2, pag. 58-69, Ian.-Feb. 1998c.

PERKINS, C.E. și ROYER, E.: "The Ad Hoc On-Demand Distance-Vector Protocol", in *Ad Hoc Networkin*g, edited by C. Perkins, Boston: Addison-Wesley, 2001.

PERKINS, C.E. și ROYER, E.: "Ad-hoc On-Demand Distance Vector Routing", *Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications*, IEEE, pag. 90-100, 1999.

PERLMAN, R.: *Interconnections*, ediția a doua, Boston: Addison-Wesley, 2000.

PERLMAN, R.: *Network Layer Protocols with Byzantine Robustness*, Ph.D. thesis, M.I.T., 1988.

PERLMAN, R. și KAUFMAN, C.: "Key Exchange in IPsec", *IEEE Internet Computing*, vol. 4, pag. 50-56, Nov.-Dec. 2000.

PETERSON, L.L. și DAVIE, B.S.: *Computer Networks: A Systems Approach*, San Francisco: Morgan Kaufmann, 2000.

PETERSON, W.W. și BROWN, D.T.: "Cyclic Codes for Error Detection", *Proc. IRE*, vol. 49, pag. 228-235, Ian. 1961.

PICKHOLTZ, R.L., SCHILLING, D.L. și MILSTEIN, L.B.: "Theory of Spread Spectrum Communication—A Tutorial", *IEEE Trans. on Commun.*, vol. COM-30, pag. 855-884, Mai 1982.

PIERRE, G., KUZ, I., VAN STEEN, M., TANENBAUM, A.S.: "Differentiated Strategies for Replicating Web Documents", *Computer Commun.*, vol. 24, pag. 232-240, Feb. 2001.

PIERRE, G., VAN STEEN, M. și TANENBAUM, A.S.: "Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Trans. on Computers*, vol. 51, Iunie 2002.

PISCITELLO, D.M. și CHAPIN, A.L.: *Open Systems Networking: TCP/IP and OSI*, Boston: Addison-Wesley, 1993.

PITT, D.A.: "Bridging—The Double Standard", *IEEE Network Magazine*, vol. 2, pag. 94- 95, Ian. 1988.

PIVA, A., BARTOLINI, F. și BARNI, M.: "Managing Copyrights in Open Networks", *IEEE Internet Computing*, vol. 6, pag. 18-26, Mai-Iunie 2002.

POHLMANN, N.: *Firewall Systems*, Bonn, Germany: MITP-Verlag, 2001.

PUZMANOVA, R.: *Routing and Switching: Time of Convergence*?, London: Addison-Wesley, 2002.

RABINOVICH, M. și SPATSCHECK, O,: *Web Caching and Replication*, Boston: Addison-Wesley, 2002.

RAJU, J. și GARCIA-LUNA-ACEVES, J.J.: "Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks", *ACM Computer Communications Review*, vol. 31, October 2001.

RAMANATHAN, R. și REDI, J.: "A Brief Overview of Ad Hoc Networks: Challenges and Directions", *IEEE Commun. Magazine*, 50th Anniversary Issue, pag. 20-22, Mai 2002.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R. și SHENKER, S.: "A Scalable Content-Addressable Network", *Proc. SIGCOMM '01 Conf.*, ACM, pag. 1161- 1172, 2001.

RIVEST, R.L.: "The MD5 Message-Digest Algorithm", RFC 1320, Aprilie 1992.

RIVEST, R.L. și SHAMIR, A.: "How to Expose an Eavesdropper", *Commun. of the ACM*, vol. 27, pag. 393-395, Aprilie 1984.

RIVEST, R.L., SHAMIR, A. și ADLEMAN, L.: "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM*, vol. 21, pag. 120-126, Feb. 1978.

ROBERTS, L.G.: "Dynamic Allocation of Satellite Capacity through Packet Reservation", *Proc. NCC*, AFIPS, pag. 711-716, 1973.

ROBERTS, L.G.: "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", *Proc. Spring Joint Computer Conference*, AFIPS, pag. 295-298, 1972.

ROBERTS, L.G.: "Multiple Computer Networks and Intercomputer Communication", *Proc. First Symp. on Operating Systems Prin.*, ACM, 1967.

ROSE, M.T.: *The Simple Book*, Englewood Cliffs, NJ: Prentice Hall, 1994.

ROSE, M.T.: *The Internet Message*, Englewood Cliffs, NJ: Prentice Hall, 1993.

ROSE, M.T. și MCCLOGHRIE, K.: *How to Manage Your Network Using SNMP*, Englewood Cliffs, NJ: Prentice Hall, 1995.

ROWSTRON, A. și DRUSCHEL, P.: "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility", *Proc. 18th Symp. on Operating Systems Prin.*, ACM, pag. 188-201, 2001a.

ROWSTRON, A. și DRUSCHEL, P.: "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms*, ACM/IFIP, 2001b.

ROYER, E.M. și TOH, C.-K.: "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Commun. Magazine*, vol. 6, pag. 46-55, Aprilie 1999.

RUIZ-SANCHEZ, M.A., BIERSACK, E.W. și DABBOUS, W.: "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine*, vol. 15, pag. 8-23, Martie-Aprilie 2001.

SAIRAM, K.V.S.S.S.S., GUNASEKARAN, N. și REDDY, S.R.: "Bluetooth in Wireless Communication", *IEEE Commun. Mag*., vol. 40, pag. 90-96, Iunie 2002.

SALTZER, J.H., REED, D.P. și CLARK, D.D.: "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems*, vol. 2, pag. 277-288, Nov. 1984.

SANDERSON, D.W. și DOUGHERTY, D.: *Smileys*, Sebastopol, CA: O'Reilly, 1993.

SARI, H., VANHAVERBEKE, F. și MOENECLAEY, M.: "Extending the Capacity of Multiple Access Channels", *IEEE Commun. Magazine*, vol. 38, pag. 74-82, Ian. 2000.

SARIKAYA, B.: "Packet Mode in Wireless Networks: Overview of Transition to Third Generation", *IEEE Commun. Magazine*, vol. 38, pag. 164-172, Sept. 2000.

SCHNEIER, B.: *Secrets and Lies*, New York: Wiley, 2000.

SCHNEIER, B.: *Applied Cryptography*, ediția a doua, New York: Wiley, 1996.

SCHNEIER, B.: *E-Mail Security*, New York: Wiley, 1995.

SCHNEIER, B.: "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]", *Proc. of the Cambridge Security Workshop*, Berlin: Springer-Verlag LNCS 809, pag. 191-204, 1994.

SCHNORR, C.P.: "Efficient Signature Generation for Smart Cards", *Journal of Cryptology*, vol. 4, pag. 161-174, 1991.

SCHOLTZ, R.A.: "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun*., vol. COM-30, pag. 822-854, Mai 1982.

SCOTT, R.: "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia*, vol. 9, pag. 75-90, Ian. 1985.

SEIFERT, R.: *The Switch Book*, Boston: Addison-Wesley, 2000.

SEIFERT, R.: *Gigabit Ethernet*, Boston: Addison-Wesley, 1998.

SENN, J.A.: "The Emergence of M-Commerce", *Computer*, vol. 33, pag. 148-150, Dec. 2000.

SERJANTOV, A.: "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems*, Berlin: Springer-Verlag LNCS, 2002.

SEVERANCE, C.: "IEEE 802.11: Wireless Is Coming Home", *Computer*, vol. 32, pag. 126-127, Nov. 1999.

SHAHABI, C., ZIMMERMANN, R., FU, K. și YAO, S.-Y.D.: "YIMA: A Second-Generation Continuous Media Server", *Computer*, vol. 35, pag. 56-64, Iunie 2002.

SHANNON, C.: "A Mathematical Theory of Communication", *Bell System Tech. J*., vol. 27, pag. 379-423, Iulie 1948; și pag. 623-656, Oct. 1948.

SHEPARD, S.: *SONET/SDH Demystified*, New York: McGraw-Hill, 2001.

SHREEDHAR, M. și VARGHESE, G.: "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.*, ACM, pag. 231-243, 1995.

SKOUDIS, E.: *Counter Hack*, Upper Saddle River, NJ: Prentice Hall, 2002.

SMITH, D.K. și ALEXANDER, R.C.: *Fumbling the Future*, New York: William Morrow, 1988.

SMITH, R.W.: *Broadband Internet Connections*, Boston: Addison Wesley, 2002.

SNOEREN, A.C. și BALAKRISHNAN, H.: "An End-to-End Approach to Host Mobility, " *Int'l Conf. on Mobile Computing and Networking* , ACM, pag. 155-166, 2000.

SOBEL, D.L.: "Will Carnivore Devour Online Privacy", *Computer*, vol. 34, pag. 87-88, Mai 2001.

SOLOMON, J.D.: *Mobile IP: The Internet Unplugged*, Upper Saddle River, NJ: Prentice Hall, 1998.

SPOHN, M. și GARCIA-LUNA-ACEVES, J.J.: "Neighborhood Aware Source Routing", *Proc. ACM MobiHoc 2001*, ACM, pag. 2001.

SPURGEON, C.E.: *Ethernet: The Definitive Guide*, Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W.: *Data and Computer Communications*, ediția a șasea, Upper Saddle River, NJ: Prentice Hall, 2000.

STEINMETZ, R. și NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 1: Media Coding and Content Processing*, Upper Saddle River, NJ: Prentice Hall, 2002.

STEINMETZ, R. și NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 2: Media Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2003a.

STEINMETZ, R. și NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 3: Documents, Security, and Applications*, Upper Saddle River, NJ: Prentice Hall, 2003b.

STEINER, J.G., NEUMAN, B.C. și SCHILLER, J.I.: "Kerberos: An Authentication Service for Open Network Systems", *Proc. Winter USENIX Conf.*, USENIX, pag. 191- 201, 1988.

STEVENS, W.R.: *UNIX Network Programming, Volume 1: Networking APIs - Sockets and XTI*, Upper Saddle River, NJ: Prentice Hall, 1997.

STEVENS, W.R.: *TCP/IP Illustrated*, Vol. 1, Boston: Addison-Wesley, 1994.

STEWART, R. și METZ, C.: "SCTP: New Transport Protocol for TCP/IP", *IEEE Internet Computing*, vol. 5, pag. 64-69, Nov.-Dec. 2001.

STINSON, D.R.: *Cryptography Theory and Practice*, ediția a doua, Boca Raton, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M.F. și BALAKRISHNAN, H.: "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM '01 Conf.*, ACM, pag. 149-160, 2001.

STRIEGEL, A. și MANIMARAN, G.: "A Survey of QoS Multicasting Issues", *IEEE Commun. Mag.*, vol. 40, pag. 82-87, Iunie 2002.

STUBBLEFIELD, A., IOANNIDIS, J. și RUBIN, A.D.: "Using the Fluhrer, Mantin și Shamir Attack to Break WEP", *Proc Network and Distributed Systems Security Symp.*, ISOC, pag. 1-11, 2002.

SUMMERS, C.K.: *ADSL: Standards, Implementation, and Architecture*, Boca Raton, FL: CRC Press, 1999.

SUNSHINE, C.A. și DALAL, Y.K.: "Connection Management in Transport Protocols", *Computer Networks*, vol. 2, pag. 454-473, 1978.

TANENBAUM, A.S.: *Modern Operating Systems*, Upper Saddle River, NJ: Prentice Hall, 2001.

TANENBAUM, A.S. și VAN STEEN, M.: *Distributed Systems: Principles and Paradigms*, Upper Saddle River, NJ: Prentice Hall, 2002.

TEGER, S. și WAKS, D.J.: "End-User Perspectives on Home Networking", *IEEE Commun. Magazine*, vol. 40, pag. 114-119, Aprilie 2002.

THYAGARAJAN, A.S. și DEERING, S.E.: "Hierarchical Distance-Vector Multicast Routing for the MBone", *Proc. SIGCOMM '95 Conf.*, ACM, pag. 60-66, 1995.

TITTEL, E., VALENTINE, C., BURMEISTER, M. și DYKES, L.: *Mastering XHTML*, Alameda, CA: Sybex, 2001.

TOKORO, M. și TAMARU, K.: "Acknowledging Ethernet", *Compcon*, IEEE, pag. 320-325, Fall 1977.

TOMLINSON, R.S.: "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, pag. 11-23, 1975.

TSENG, Y.-C., WU, S.-L., LIAO, W.-H. și CHAO, C.-M.: "Location Awareness in Ad Hoc Wireless Mobile Networks", *Computer*, vol. 34, pag. 46-51, 2001.

TUCHMAN, W.: "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum*, vol. 16, pag. 40-41, Iulie 1979.

TURNER, J.S.: "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine*, vol. 24, pag. 8-15, Oct. 1986.

VACCA, J.R.: *I-Mode Crash Course*, New York: McGraw-Hill, 2002.

VALADE, J.,: *PHP & MySQL for Dummies*, New York: Hungry Minds, 2002.

VARGHESE, G. și LAUCK, T.: "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.*, ACM, pag. 25-38, 1987.

VARSHNEY, U., SNOW, A., MCGIVERN, M. și HOWARD, C.: "Voice over IP", *Commun. of the ACM*, vol. 45, pag. 89-96, 2002.

VARSHNEY, U. și VETTER, R.: "Emerging Mobile and Wireless Networks", *Commun. of the ACM*, vol. 43, pag. 73-81, Iunie 2000.

VETTER, P., GODERIS, D., VERPOOTEN, L. și GRANGER, A.: "Systems Aspects of APON/VDSL Deployment", *IEEE Commun. Magazine*, vol. 38, pag. 66-72, Mai 2000.

WADDINGTON, D.G. și CHANG, F.: "Realizing the Transition to IPv6", *IEEE Commun. Mag.*, vol. 40, pag. 138-148, Iunie 2002.

WALDMAN, M., RUBIN, A.D. și CRANOR, L.F.: "Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System", *Proc. Ninth USENIX Security Symp.*, USENIX, pag. 59-72, 2000.

WANG, Y. și CHEN, W.: "Supporting IP Multicast for Mobile Hosts", *Mobile Networks and Applications*, vol. 6, pag. 57-66, Ian.-Feb. 2001.

WANG, Z.: *Internet QoS*, San Francisco: Morgan Kaufmann, 2001.

WARNEKE, B., LAST, M., LIEBOWITZ, B. și PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *Computer*, vol. 34, pag. 44-51, Ian. 2001.

WAYNER, P.: *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking*, ediția a doua, San Francisco: Morgan Kaufmann, 2002.

WEBB, W.: "Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment", *IEEE Commun. Magazine*, vol. 39, pag. 115-121, Sept. 2001.

WEISER, M.: "Whatever Happened to the Next Generation Internet?", *Commun. of the AC*M, vol. 44, pag. 61-68, Sept. 2001.

WELTMAN, R. și DAHBURA, T.: *LDAP Programming with Java*, Boston: Addison-Wesley, 2000.

WESSELS, D.: *Web Caching*, Sebastopol, CA: O'Reilly, 2001.

WETTEROTH, D.: *OSI Reference Model for Telecommunications*, New York: McGraw-Hill, 2001.

WILJAKKA, J.: "Transition to Ipv6 in GPRS and WCDMA Mobile Networks", *IEEE Commun. Magazine*, vol. 40, pag. 134-140, Aprilie 2002.

WILLIAMSON, H.: *XML: The Complete Reference*, New York: McGraw-Hill, 2001.

WILLINGER, W., TAQQU, M.S., SHERMAN, R. și WILSON, D.V.: "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proc. SIGCOMM '95 Conf.*, ACM, pag. 100-113, 1995.

WRIGHT, D.J.: *Voice over Packet Networks*, New York: Wiley, 2001.

WYLIE, J., BIGRIGG, M.W., STRUNK, J.D., GANGER, G.R., KILICCOTE, H. și KHOSLA, P.K.: "Survivable Information Storage Systems", *Computer*, vol. 33, pag. 61-68, Aug. 2000.

XYLOMENOS, G., POLYZOS, G.C., MAHONEN, P. și SAARANEN, M.: "TCP Performance Issues over Wireless Links" , *IEEE Commun. Magazine*, vol. 39, pag. 52-58, Aprilie 2001.

YANG, C.-Q. și REDDY, A.V.S.: "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", *IEEE Network Magazine*, vol. 9, pag. 34-45, Iulie-Aug. 1995.

YUVAL, G.: "How to Swindle Rabin", *Cryptologia*, vol. 3, pag. 187-190, Iulie 1979.

ZACKS, M.: "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing*, vol. 5, pag. 8-9, Nov.-Dec. 2001.

ZADEH, A.N., JABBARI, B., PICKHOLTZ, R. și VOJCIC, B.: "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag*., vol. 40, pag. 149-157, Iunie 2002.

ZHANG, L.: "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine*, vol. 2, pag. 44-48, Ian.-Feb. 1988.

ZHANG, L.: "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine*, vol. 7, pag. 8-18, Sept.-Oct. 1993.

ZHANG, Y. și RYU, B.: "Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications*, vol. 6, pag. 81- 94, Ian.-Feb. 2001.

ZIMMERMANN, P.R.: *The Official PGP User's Guide*, Cambridge, MA: M.I.T. Press, 1995a.

ZIMMERMANN, P.R.: *PGP: Source Code and Internals*, Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Boston: Addison-Wesley, 1949.

ZIV, J. și LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory*, vol. IT-23, pag. 337-343, Mai 1977.

INDEX

A

AAL, 57, 58, 442
ADC, 603
ALOHA, 226, 227, 228, 229, 230, 231, 232, 233, 236, 302, 303, 304
AMPS, 138, 140, 159
ANSI, 66, 67, 209
ANSNET, 50
ARP, 402, 404, 405, 414, 423, 428, 429
ARPANET, 37, 47, 48, 49, 50, 68, 84, 322, 324, 391, 511, 518, 522, 530, 536
ARQ, 187
ATM, 44, 57, 58, 71, 130, 311, 360, 376, 383, 384, 426, 429, 442, 457, 461, 499, 500

B

BGP, 411, 412, 416, 423
BOC, 110
BOOTP, 402, 405

C

CDPD, 329
CIDR, 395, 396, 415
Consiliul Arhitecturii Internet, 68
CSMA, 231, 232, 233, 235, 242, 243, 251, 254, 302, 304, 305
CSMA/CD, 232, 233, 254, 302, 304, 305

D

DCS, 533
DES, 660, 661, 662, 666, 667, 668, 671, 674, 679, 711, 716
DNS, 39, 48, 404, 416, 521, 522, 524, 525, 526, 527, 529, 533, 534, 551, 552, 558, 559
domeniu, 7, 24, 41, 67, 113, 124, 125, 160, 218, 225, 241, 242, 254, 315, 335, 336, 416, 522, 523, 524, 525, 526, 528, 571, 648, 711
DSS, 677

F

FAQ, 547
FCC, 97, 103, 138
FDDI, 403, 405
FDM, 101, 123, 124, 125, 126, 140, 159, 163, 224, 225, 238, 302
FTP, 39, 51, 478, 539, 559, 560, 694

G

Gopher, 559, 560
GSM, 304

H

HDLC, 70, 209, 211, 212, 214, 215, 217, 219, 220
HDTV, 620, 626
HTML, 538, 559, 564, 565, 566, 567, 568, 569, 570, 599
HTTP, 40, 559, 579, 580, 581, 582, 584, 585, 586, 588, 589, 592, 614, 616

I

IAB, 68, 69

IBM, 16, 43, 50, 64, 83, 209, 411, 660, 661, 662
ICMP, 402, 413, 416, 419, 420, 421, 423, 493
IDEA, 714, 715
IEEE, 16, 68, 69, 242, 243, 248, 288, 289, 416, 662
IEEE 802, 16, 242, 243, 288
IGMP, 413, 416
IMP, 46, 47, 49
IMTS, 138
IP, 34, 37, 38, 39, 40, 41, 42, 43, 44, 48, 50, 57, 68, 72, 74, 213, 214, 216, 220, 329, 375, 376, 380, 381, 383, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 402, 403, 404, 405, 407, 410, 412, 413, 414, 415, 416, 417, 418, 419, 422, 423, 425, 427, 428, 429, 442, 471, 477, 478, 479, 480, 481, 482, 484, 488, 495, 496, 503, 508, 509, 516, 522, 524, 525, 526, 527, 528, 551, 559, 694
IPv4, 415, 416, 417, 418, 419, 421, 422, 429
IPX, 214, 329, 375, 376
ISDN, 130
IS-IS, 329
ISO, 34, 35, 66, 67, 68, 69, 209, 329, 522, 566, 622
ITU, 65, 66, 68, 91, 622
ITU-R, 65
ITU-T, 65, 66, 68
IXC, 110, 111

J

JPEG, 538, 567, 622, 623, 624, 625, 626, 627

K

KDC, 701, 707, 708, 709, 710
Kerberos, 709, 710, 711

L

LAN, 16, 18, 21, 24, 48, 73, 89, 212, 223, 224, 225, 226, 227, 232, 238, 241, 242, 249, 250, 253, 254, 286, 287, 288, 289, 290, 291, 292, 302, 304, 305, 311, 325, 329, 335, 392, 403, 409, 413, 414, 425, 429
LAP, 209
LATA, 110
LCP, 213, 214, 215, 216
LEC, 110, 111
LLC, 287

M

MAC, 223, 224, 232, 233, 241, 248, 287, 292, 383
MACA, 242, 243, 302, 304
MACAW, 242, 243, 302
MAN, 16, 224, 225, 335, 518
MD5, 678, 679, 682, 714, 715, 716
MIME, 536, 537, 538, 585, 594, 595, 597, 598, 600, 601, 602, 610, 714
Mosaic, 51, 548, 549
MPEG, 538, 539, 622, 625, 626
MTSO, 139, 140
MTU, 480

N

NAK, 203, 219
NAP, 50
NCP, 213, 214, 215, 216, 375
NIC, 391
NIST, 68, 677, 678, 679
NNTP, 39, 560
Novell, 329, 375
NSA, 661, 662, 677, 679
NSAP, 442, 443
NSFNET, 49, 50, 52, 329
NTSC, 619, 620, 623, 625, 626

O

OAM, 130
OC, 132
OSPF, 329, 406, 407, 408, 409, 410, 411, 413, 416, 423

P

PAL, 619, 620, 623, 626
PCM, 126, 127, 130, 161
PCS, 163
PEM, 712, 716
PGP, 712, 713, 714, 715, 716
POP, 110
POP3, 543
PPP, 70, 213, 214, 215, 217, 220
PSTN, 107
PTT, 64

Q

QAM, 115

R

RARP, 402, 405, 423, 428, 500

RFC, 69, 214, 216, 390, 396, 404, 405, 406, 407, 412, 413, 416, 419, 471, 477, 483, 534, 535, 536, 537, 538, 539, 543, 714, 716
RSVP, 368

S

SABME, 211
SAP, 74, 442
SDH, 129, 131, 132
SDLC, 209, 211, 217
SECAM, 619, 620, 623
SGML, 538
SHA, 678, 679
SIPP, 416
SLIP, 213
Smiley, 529
SMTP, 39, 540, 541, 542, 543
SNA, 209
SNRME, 211
SONET, 44, 124, 129, 130, 131, 132, 162, 214, 215, 621
SPE, 131, 132

T

TCP, 34, 37, 38, 39, 40, 41, 42, 43, 44, 48, 50, 57, 68, 72, 74, 213, 375, 379, 381, 390, 412, 416, 418, 419, 436, 444, 471, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 499, 500, 503, 508, 509, 514, 517, 522, 540, 541, 542, 551, 559, 694
TDM, 124, 129, 130, 159, 163, 225, 238, 302, 383
TPDU, 434, 435, 436, 437, 445, 446, 447, 448, 449, 450, 451, 453, 454, 455, 456, 457, 458, 460, 499, 500, 502, 503, 505, 507, 508, 509, 517
TSAP, 442, 443, 444, 445, 460, 461, 478, 481

U

UDP, 39, 74, 239, 390, 405, 416, 418, 419, 471, 496, 497, 500, 503, 516, 522, 528, 694
URI, 560
URL, 551, 558, 559, 560, 567, 585
UTP, 83

V

V.24, 65
V.32, 116
VSAT, 101, 102

W

WAN, 17, 23, 24, 168, 223, 311, 335, 381, 405, 432
WDM, 125, 126, 159
WDMA, 238, 240, 304
Web, 40, 51, 503, 521, 538, 548, 549, 550, 552, 558, 559, 560, 564, 565, 567, 568, 569, 571, 584, 585, 586, 617, 694
WWW, 51, 179, 497, 548, 551, 559, 565

X

X.400, 530, 533

DESPRE AUTOR

Andrew S. Tanenbaum este absolvent al M.I.T. și a obținut titlul de doctor la University of California din Berkeley. În acest moment este profesor la Vrije Universiteit din Amsterdam, Olanda, unde este conducător al grupului care se ocupă de sistemele de calculatoare (Computer Systems Group). De asemenea, este decan al Școlii Avansate de Calculatoare și Prelucrarea Imaginilor (Advanced School for Computing and Imaging), un institut de învățământ interuniversitar care se ocupă cu studiul sistemelor paralele, distribuite și de prelucrare a imaginilor avansate. Cu toate acestea, el încearcă din răsputeri să evite să devină un birocrat.

În trecut, a efectuat cercetări în domeniul compilatoarelor, al sistemelor de operare, al rețelelor și al sistemelor distribuite locale. Domeniul în care efectuează acum cercetări este cel al sistemelor distribuite globale care pot fi scalate până la un miliard de utilizatori. Aceste cercetări, efectuate împreună cu prof. Maarten van Steen sunt prezentate la *www.cs.vu.nl/globe*. Împreună, toate aceste proiecte de cercetare au dus la scrierea a peste 100 de articole publicate în diverse reviste sau prezentate la diferite conferințe și la scrierea a cinci cărți.

Prof. Tanenbaum a creat, de asemenea, numeroase produse software. A fost principalul arhitect al produsului Amsterdam Compiler Kit, un set de unelte folosit pe scară largă pentru scrierea de compilatoare portabile. De asemenea, este creatorul sistemului de operare MINIX, o clonă UNIX destinată utilizării la orele de laborator ale studenților. Acest sistem de operare a fost sursa de inspirație și baza sistemului de operare LINUX. Împreună cu doctoranzii săi și cu mai mulți programatori, a contribuit la proiectarea sistemului de operare distribuit Amoeba, un sistem foarte performant bazat pe un micronucleu. MINIX și Amoeba sunt acum disponibile gratuit pe Internet.

Doctoranzii săi și-au urmat drumul spre glorie după obținerea titlurilor de doctori. E foarte mândru de ei. În această privință, el se aseamănă cu o cloșcă.

Prof. Tanenbaum este membru ACM, IEEE, precum și al Academiei Regale de Arte și Științe din Olanda (Royal Netherlands Academy of Arts and Sciences). De-a lungul timpului a obținut numeroase distincții: ACM Karl V. Karlstrom Outstanding Educator Award în 1994, ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education în 1997 și Texty în 2002. Aceasta din urmă este acordată pentru excelență în scrierea cărților. De asemenea, numele său și o scurtă prezentare apare în cartea *Who's Who in the World*. Pagina sa Web este disponibilă la adresa *http://www.cs.vu.nl/~ast/*.